Globally Asynchronous and Locally Synchronous

Project Report submitted in partial fulfillment of the

requirement for the degree of

Bachelor of Technology

in

Computer Science & Engineering

under the Supervision of

Dr. Vivek Sehgal

By

Rachit Agarwal
111287

to

Jaypee University of Information and Technology

Waknaghat, Solan – 173234, Himachal Pradesh

Certificate

This is to certify that project report entitled "Globally Asynchronous and Locally Synchronous", submitted by Rachit Agarwal in partial fulfillment for the award of degree of Bachelor of Technology in Computer Science & Engineering to Jaypee University of Information Technology, Waknaghat, Solan has been carried out under my supervision.

This work has not been submitted partially or fully to any other University or Institute for the award of this or any other degree or diploma.

Date: **Dr. Vivek Sehgal**

Associate Professor

Acknowledgement

There are many people who are associated with this project directly or indirectly whose help and timely suggestions are highly appreciable for completion of this project. First of all, I would like to thank Prof. Dr. SP Ghrera, Head, Department of Computer Science Engineering for his kind support and constant encouragements, valuable discussions which is highly commendable.

I would like to express my sincere gratitude to my supervisor Dr. Vivek Sehgal, for his super vision, encouragement, and support which has been instrumental for the success of this project. It was an invaluable experience for me to be one of his students. Because of him, I have gained a careful research attitude.

Lastly, I would also like to thank my parents for their love and affection and especially their courage which inspired me and made me to believe in myself.

Date:

Rachit Agarwal
Roll No. 111287

Table of Content

Abbreviations

2-D	2-Dimensional
AHB	Advanced High-performance Bus
AMBA	Advanced Microcontroller Bus Architecture
APB	Advanced Peripheral Bus
ASB	Advanced System Bus
ASIC	Application Specific Integrated Circuits
ATL	Asynchronous Transfer Latency
BE	Best-Effort
CDMA	Code-Division Multiple-Access
D-FF	D-Flip-Flop
DI	Delay-Insensitive
DS-CDMA	Direct-Sequence CDMA
DSM	Deep Sub-Micron
FF	Flip-Flop
FPGA	Field-Programmable Gate Array
GALS	Globally-Asynchronous Locally-Synchronous
HDL	Hardware Description Language
IC	Integrated Circuit
MTBF	Mean Time Between Failure
NoC	Network-on-Chip
OSI	Open Systems Interconnection
PCB	Printed Circuit Board
PCC	Packet Connected Circuit
PCI	Peripheral Component Interconnect
PTL	Packet Transfer Latency
QoS	Quality of Service
RTL	Register-Transfer Level
SoC	System-on-Chip

STL	Synchronous Transfer Latency
VCI	Virtual Component Interface
VHDL	VHSIC Hardware Description Language
VHSIC	Very High Speed Integrated Circuit

List of Figures

Abstract

This report addresses the aspect of designing on-chip communication network. This is about applying Globally-Asynchronous Locally-Synchronous (GALS) communication scheme into Network-on-Chip (NoC).

GALS scheme is applied in the NoC designs presented in this report by applying synchronous style in the communications between network nodes and their attached function hosts while applying asynchronous style in the communication among network nodes.

The Network-on-Chip (NoC) concept has recently become a widely discussed technique for handling the large on-chip communication requirements of complex System-on-Chip (SOC) designs. A traditional bus-based interconnection scheme does not scale well to very large SOCs because many Intellectual Property (IP) blocks must contend with each other to communicate over the shared bus. In contrast, an on-chip network uses the packet-switching paradigm to route information between IP blocks and it can be scaled up to achieve a very large total aggregate bandwidth within the chip.

A packet switched Network-on-Chip (NOC) that applies the GALS technique is realized in NS2. The realized NOC supports the GALS communication scheme by applying both synchronous and asynchronous designs. A six-node GALS on-chip network is modeled and simulated. The characteristics of the GALS NOC are examined by making the timing diagrams of the respective synchronous and asynchronous designs. The different 2-D topologies are also shown.

CHAPTER 1

1.1 Introduction

Communications play a fundamental and crucial role for the development of human society in every aspect because better communications facilitate better understanding and cooperation between individuals, which in turn facilitate the achievements and development in society. As the society is continuously growing and developing, the need for cooperation and development expands to global level. Therefore, communication plays more and more important role of this globalization process, and the need for effective communications in all kinds of ways become higher and higher.

The same truth also applies to on-chip systems, which means that the quality of communication in an on-chip system prominently affects system performance. As the complexity of an on-chip system keeps growing, the communication among functional hosts in the system becomes a non-trivial issue to deal with. Therefore, with interest in on-chip communication, I started my work on this topic.

Currently, silicon chips which contain thousands of millions of transistors with 45nm feature size are already available on market, e.g. Intel Pentium processor. According to the report from International Technology Roadmap for Semiconductors (ITRS) in 2007, a single semi-conductor chip will contain multi-billion transistors with feature sizes around 22nm and clock frequencies around 35GHz by the year of 2016. This growing manufacture capacity and the highly demanding applications continuously drive the complexity of a System-on- Chip (SOC) to a higher degree in terms of number of system components and functionalities. For example, Cell Broadband Engine Architecture (CBEA) jointly developed by IBM, Sony, and Toshiba, also referred as Cell processor, contains altogether 9 processing units in one chip. Furthermore, Tierra, a MIT spin-off company, released a 64-core processor called TILE 64 in 2007. As the number of system

components becomes larger, current widely applied bus structures for data transfers in an on-chip system, e.g. Core Connect, expose several disadvantages.

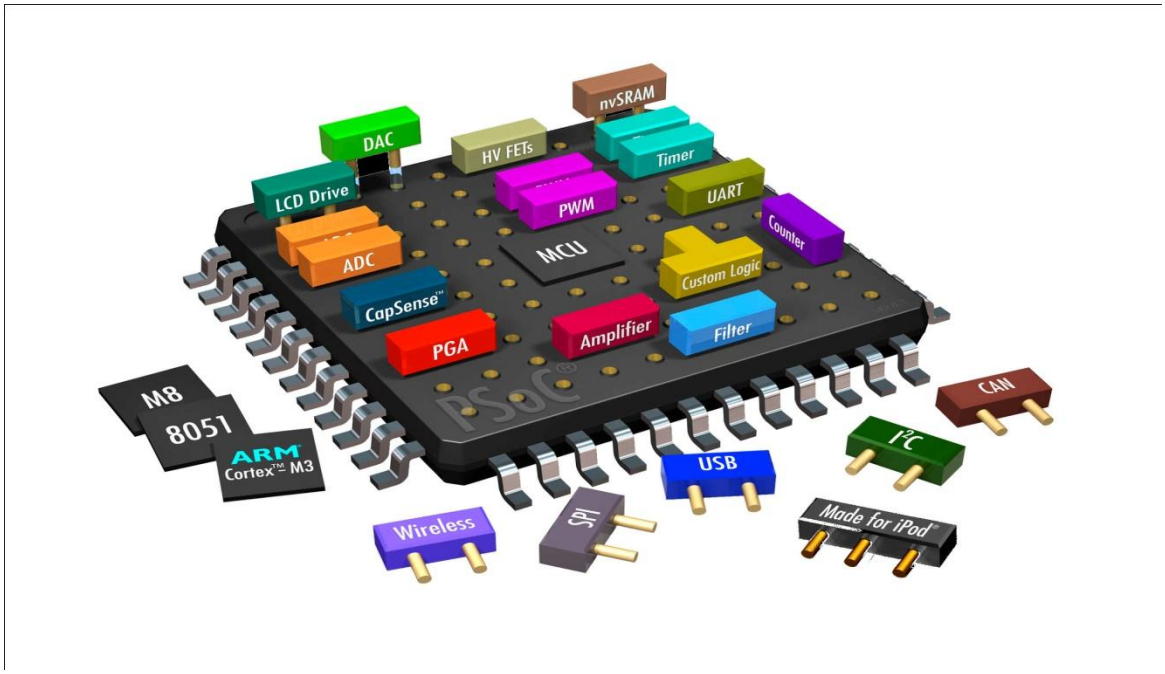

Fig. 1.1 A System-on-Chip

Two main disadvantages are bus arbitration bottleneck and bandwidth limitation. The arbitration bottleneck means that the arbitration delay will grow if the number of bus hosts increases. The bandwidth limitation refers to the fact that the data transfer bandwidth of a bus structure is shared by all hosts attached to it in a time division manner. Hence, more hosts incur a lower share of bandwidth for each one.

Another challenge that an on-chip system faces is the heterogeneous characteristics of system components. The components in a SOC may include processors for computation tasks, functional blocks for accelerating certain tasks, and the modules for communicating with the peripherals of system. The different functions among different system components naturally cause them to work in different clock rates for optimal performance. Hence, coordination and communications among those components become challenging tasks. At the same time, the issues of wire delay, on-chip noise, process variance, and power consumption in the realm of Deep Sub-Micron (DSM) technologies

also become challenging for chip design. Altogether, these challenges have brought more and more concerns on the on-chip communication issue of a SOC design.

In order to overcome the disadvantages of bus structures, the concept of Network-on-Chip (NOC) has been proposed as a solution at the beginning of 2000s, The idea of NOC is to separate the concerns of communication from computation by building on-chip communication structure with concepts adopted from computer networks. Each component of a SOC is viewed as a node of the on-chip communication network. System components communicate with each other through the on-chip network. For the challenges of multiple clock domains and DSM technology effect, Globally-Asynchronous Locally-Synchronous (GALS) scheme has been proposed as a solution. The idea of a GALS system is to partition a system into separate clock domains which run at different clock rates, and the separated domains communicate with each other in an asynchronous manner.

1.2 Evolution of On chip Networks

There are three common communication systems for system on chip i.e. point to point communication and shared bus system.

1.2.1 Point to Point Communication

Previously the designers prefer the direct point to point connection for the communication in system on chip. Here the resources or cores are allowed to communicate directly through wires which are connected to each cores. This system doesn’t need any priority providing system or arbitration unit. For a system on chip having more number of cores, this communication system requires large routing area, large routing delay and large number of pins for each core and becomes very complex in wiring point of view. When direct point to point interconnections are used for

communication, in this kind of communication system we can detect the quality of signal and delays occurred for routing. So testing of that system is a very tedious job. Due to these above problems, direct point to point interconnection system shows some disadvantages like underutilization of cores or resource can use this communication infrastructure and can give best performance as compared to other systems.

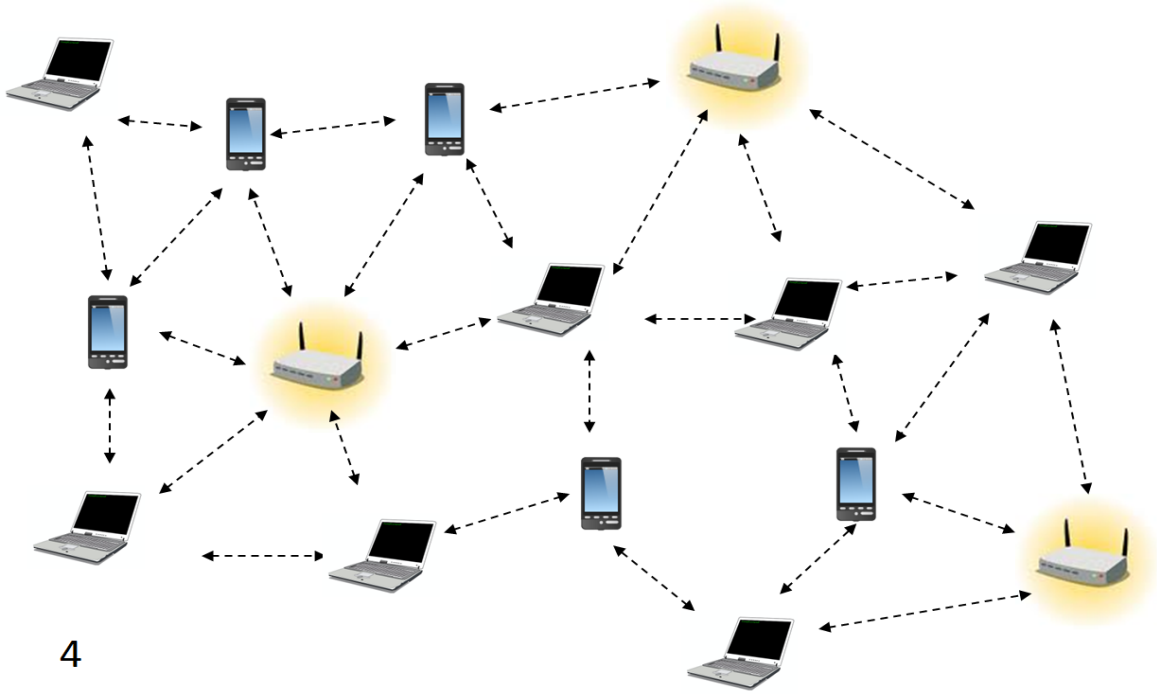

Fig 1.2 Point to point Communication

1.2.2 Shared Bus System

Most of the SoCs uses shared bus system as their inter-core communication system. Here all the cores are connected to one or more than one bus. An interface is used for the connection of the bus to the cores. In this system the communication and contention is managed by a bus arbiter system. Shared bus communication infrastructure requires less input output pins as compared to direct point to point communication system. So wiring area and cost is greatly reduced. There are different kinds of buses present in literatures such as hierarchical, segmented, pipelined buses etc. So there are many advantages of this

communication system. But still it has some disadvantages like due to contention and arbitration data movement becomes slow. In scalability point of view this system is not a fair choice as it can be scaled upto certain limits otherwise efficiency will be very bad.

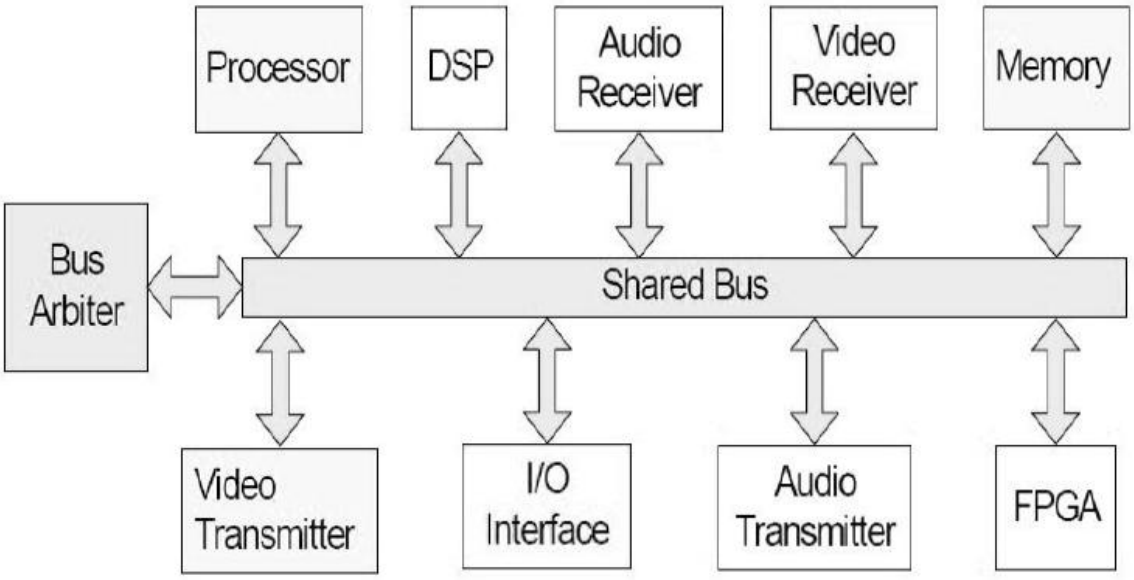

Fig 1.3 Shared Bus system for SoC

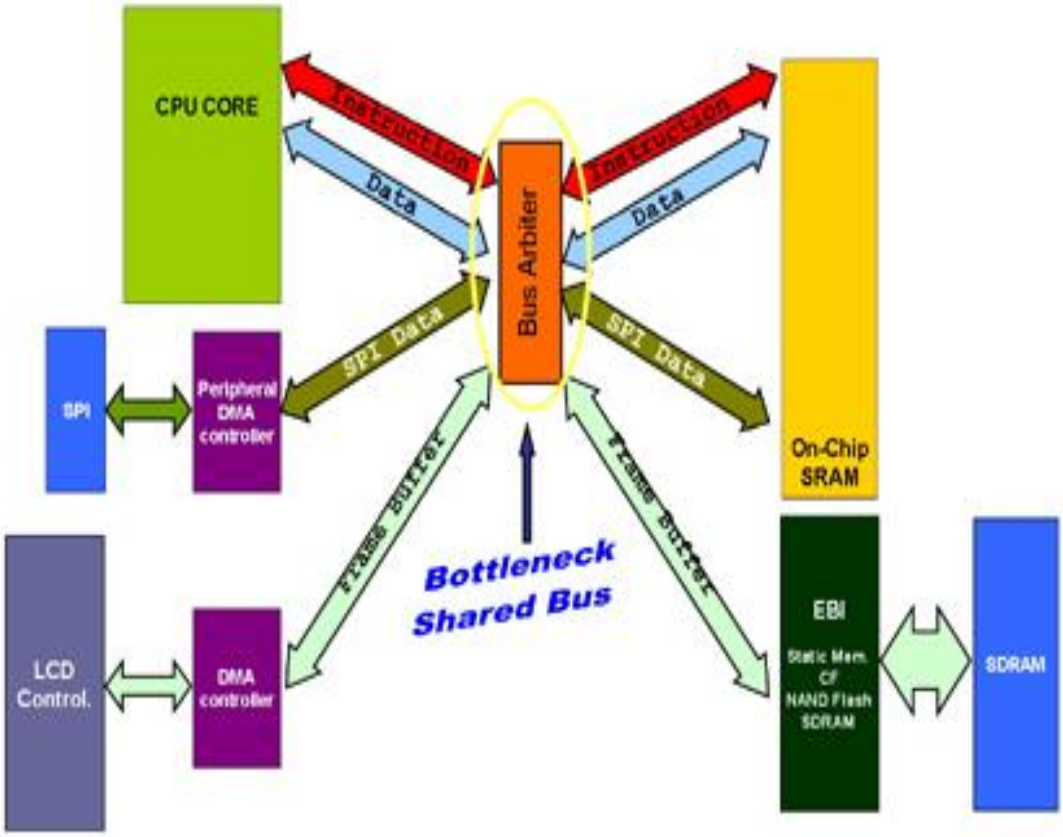

Fig. 1.4 Bus Arbitration Bottleneck

1.2.3 Network on Chip

There are many disadvantages of the above two communication systems i.e. less scalability, non-adaptive nature, underutilization of resources and less reuse factor. Many researchers proposed a communication system which can avoid above problems which is termed as Network on Chip shown in Figure 1.3. It consists of three important components i.e. Routers, Resource Network Interface (RNI) and IP cores or resources. IP cores in the NoC are connected to the network switches. RNI (Resource Network Interface) is the communication bridge between the routers and IP cores as routers and IP cores have different communication protocols. For this on chip packet switched network data is converted into some formatted packets and those packets traverse from source to destination with the help of one or more routers in the network. Scalability of this

communication system is sufficiently high. It also provides high reusability factor, less complexity and reduced cost.

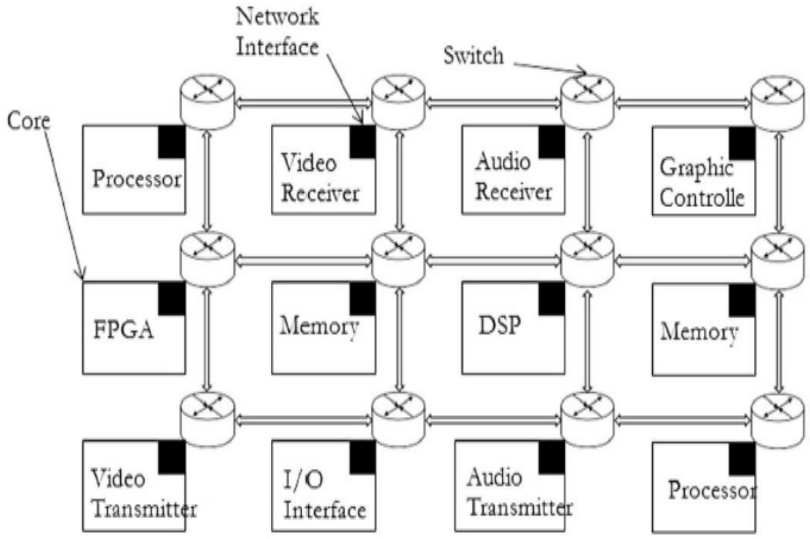

Fig. 1.5 NoC for a SoC

CHAPTER 2

2.1 Network-on-Chip Overview

The appearance of Integrated Circuit (IC) in 1959 was a milestone of the development of electronics industry. It created a productive way to manufacture large scale electronic circuits on a semiconductor device. As stated by Gordon Moore in 1965, ―the complexity for minimum component costs has increased at a rate of roughly a factor of two per year‖. This statement is known as the original formulation of Moore‘s law and often quoted as ―the number of transistors that can be placed on an IC is increasing exponentially, doubling approximately every two years. The Moore‘s Law is still valid now a days and believed to be valid until reaching the size of atoms.

Therefore, driven by the growing manufacture capacity and the growing requirement of applications, the complexity of an on-chip system is continuously growing in terms of number of transistors and functionalities. For example, Intel‘s ‘Core 2 Duo’ processor fabricated with 65nm technology process contains 291 million transistors. When the on-chip system becomes complicated, the system design methodology called orthogonalization of concerns can be applied to deal with the complexity. The communication issue is very crucial for an on-chip system to perform its tasks efficiently. Therefore, in the context of SOC design, one way of applying the methodology of concerns orthogonalization is to separate the concerns of communication from computation to enable more efficient exploration of optimal solutions on each subject.

On-chip bus structure was firstly applied to handle on-chip communications for a SoC design in 1990s. The idea of on-chip bus is derived from the bus schemes, such as Versa Module Eurocard (VME) bus and Peripheral Component Interconnect (PCI) bus, which are designed for connecting discrete devices on a Printed Circuit Board (PCB). The examples of on-chip bus structures include CoreConnect and Advanced Microcontroller Bus Architecture (AMBA). CoreConnect is a complete and versatile bus specification

which defines three types of buses: Processor Local Bus (PLB), On-chip Peripheral Bus (OPB) and Device Control Register Bus (DCR). AMBA, which is similar to CoreConnect, also specifies three kinds of buses: Advanced High-performance Bus (AHB), Advanced System Bus (ASB) and Advanced Peripheral Bus (APB). These bus structures supply many advanced features, such as split transactions and line transfers, for on-chip systems which contain a few processors.

However, bus structures have several disadvantages by the comparison of on-chip networks. The main disadvantages, bus arbitration bottleneck and bandwidth limitation are caused by the centralized and time division manner of sharing a communication channel among all the hosts of a bus. The trend of future on-chip systems is that a large number of processing units will be integrated into one system. Therefore, if a bus structure is applied in the future on-chip systems which contain a large number of components, it will suffer from the problems of arbitration delay, bandwidth limitation, and poor scalability. Hence, developing a dedicated on-chip network is the most promising solution for future on-chip communication.

2.2 Main Components of NoC

A NoC has three main and basic components i.e.

i) Network switches technically called as routers

ii) Resources or IP cores

iii) Resource to Network Interfaces (RNI).

2.2.1 Resources

In a tiled, city-block style of NoC layout, the wires and routers are placed similar to street grids of a city, while the clients (e.g., IP cores or Resources) are placed on city blocks separated by wires. The IP cores or resources can be General Purpose Processors, FPGAs, Amplifiers, ADCs, DSP, memory, Graphic controller, Mixed signal Module, RF unit, application specific hardware component, I/O controller etc. Resource must have the

same technology implementation as that of used in NoC. A designer can use own resources rather than buying from different vendors.

2.2.2 RNI

A Resource Network Interface is used to connect an IP core or resource to a router in NoCs. Like that IP cores can transmit message packets to the network switch. Resource Network Interface has two parts which are i) Resource Dependent part ii) Resource Independent part. Design of Resource independent part is done in such a way that Resource Network Interface acts as another network switch to the connected network switch. The method of designing resource independent part of RNI is general kind of procedure. For reusability point of view resource dependent part should be connected to the resource having homogeneous property otherwise this resource dependent part will be different for all resources. The Resource dependent part of RNI has some functionalities like flit formation (flitization), deflitization and applying encoding system. The RNI has of two independent layers of OSI model i.e. i) Session layer ii) Transport layer. As per theoretical views, the session layer sets up, coordinates and terminates the conversation between the application hence acts as effective medium for connected IP cores and the transport layer ensures data transfer and operates on the network interface in the communication infrastructure. The transport layer offers a communication services to the upper layer i.e. session layer where message serves as a communication intermediate. The session layer is operated with the help of service provided by the bottom layer i.e. transport layer and connected IP cores are isolated from the communication network infrastructure.

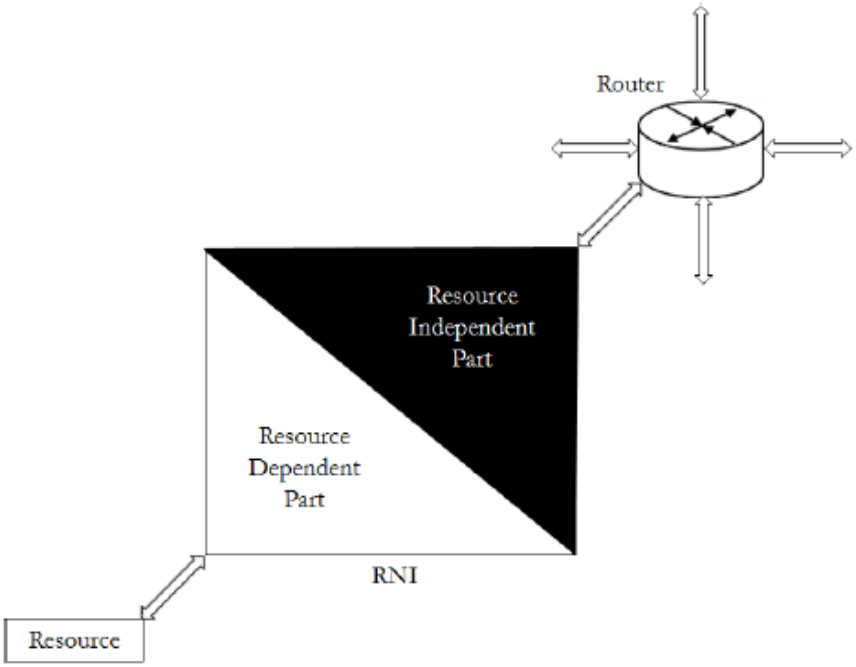

Fig 2.1 Resource Network Interface

2.2.3 Router

Router is nothing more than a switch used in the network. It is a very important part of the on chip network like any other network infrastructure. They are just like back bone of Network on Chip. In an on chip network, the primary task of a router is to transmit the incoming data to the destination IP core if the router is directly connected to the destination resource otherwise that router has to send it to another router. A router implementation is based on three layers of communication in OSI model i.e. Physical, Data link and Network layer. A designer should consider the simplicity of a router and design it like wise so that he can avoid some overheads like cost, area and power. Routing function implementation is the sole purpose of router for distributed routing. For routing purpose a router may contain a routing table which is called as table based router and that table stores the entire route. In another way router implements routing algorithm to calculate the routing path dynamically. The router used for distributed routing is very complex because it needs memory and extra logic to implement entire routing function. A

generic router consist of five ports i.e. east, west, north, south and local port and a central cross point matrix. The first four ports are used to connect to other routers and the local port is used to connect the IP core. In the router every port has an input channel and an output channel. The data packets move into the input channel of a port of router by which it is moved to the output channel of other port. The input and output channels have their own decoding logic which enhances the performance of the router. Buffers work as temporary storage of data. Here the buffering method used is store and forward. Control logic is required to make arbitration decisions. Thus, a communication is set up between the input and output ports. This connection or configuration between these ports is formed by the central cross point matrix.

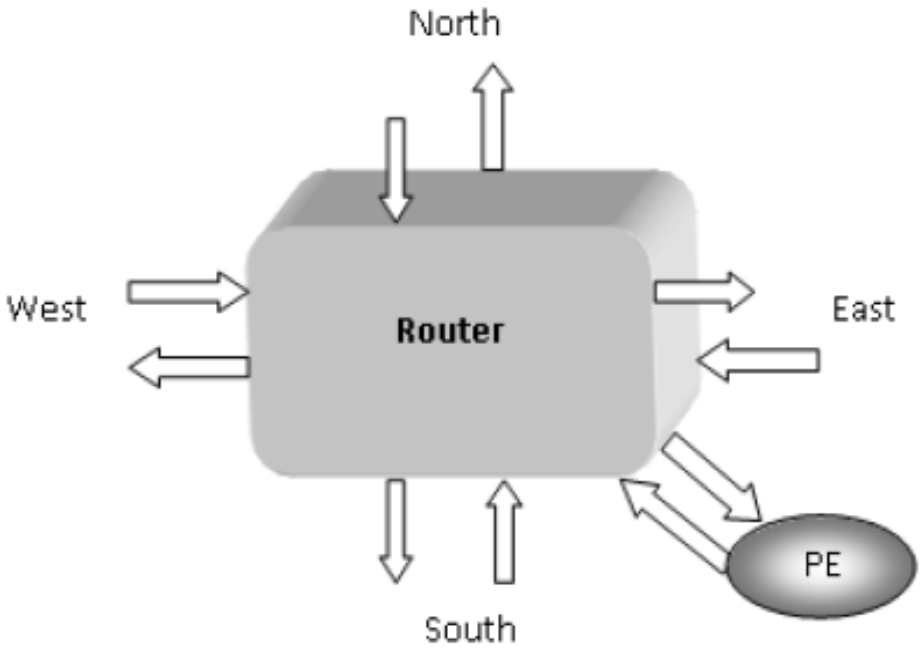

Fig 2.2 Generic Router

2.3 Network-on-Chip Design Issues

The concept of on-chip networks is derived from the well-established inter-computer networks. Therefore, taking a look at the design issues of designing computer networks is helpful for tackling design problems of NoC because they have a lot of similarities

despite of different characteristics and application environments. The Open System Interconnection (OSI) reference model is a layered description which has been used for building computer networks. Thus, the NoC design issues can be addressed according to the OSI reference model.

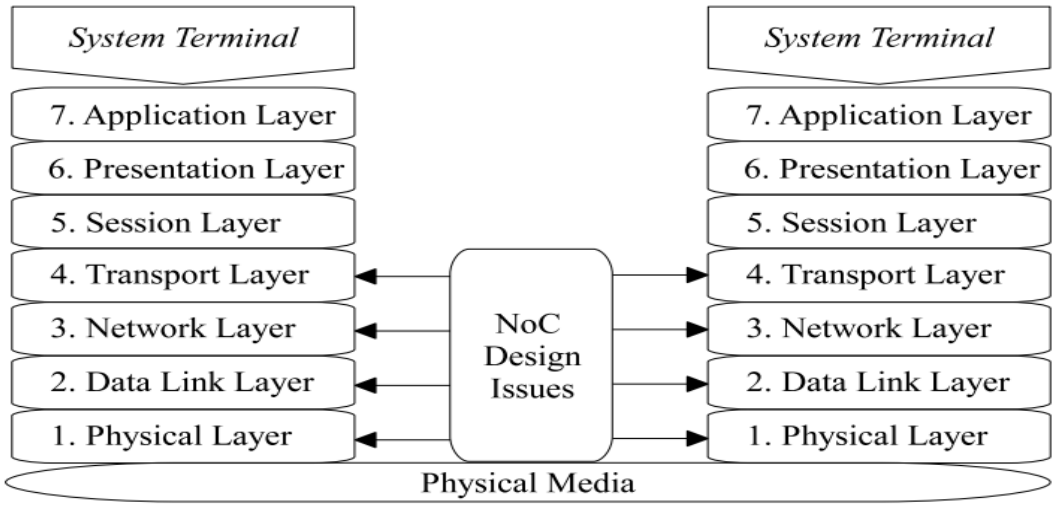

Fig 2.3 ISO Open System Interconnection Reference Model

Seven layers are defined in the OSI model. The seven layers include application layer, presentation layer, session layer, transport layer, network layer, data link layer, and physical layer. Each layer provides certain services to facilitate the communication processes in the network. The issues and challenges of designing on-chip networks will be addressed together with describing the functions of each layer in the following paragraphs of this section.

2.3.1 Physical Layer

This layer defines all the electrical and physical specifications to activate, maintain, and de-activate physical connections for data transfers. Normally, a NoC design is implemented on a silicon chip in which characteristics of the physical connection medium are determined by the manufacturing technology. As the manufacture

technology scales down to DSM domain, the on-chip physical links face the challenges of large wire delay, large power consumption, crosstalk noise, etc. Therefore, the NoC design efforts in this layer mainly concentration conquering the above mentioned challenges in physical level.

2.3.2 Data Link Layer

This layer is responsible for setting up reliable data transfers over physical links. The NoC design issues in this layer can include error detection and correction, access arbitration of physical media, and the methods of utilizing physical links. Another design issue related with this layer is the multi-clock-domain communication issue. In a large on-chip system, different functional hosts may work in different clock domains in order to achieve optimal performance; hence data transfer crossing clock domains is a design challenge. Another approach is to apply GALS scheme into on-chip networks. It means that the global links and the local links in a large on-chip system apply different communication methods to solve the multiple clock domain problem and increase data transfer reliability.

2.3.3 Network Layer

The network layer provides the means of data transfers through a network connection between a source and a destination. It should make the transport layer independent on the data routing and relay considerations. For a NoC design, the main issues to be handled in this layer include network topology and data routing.

Network topology concerns the layout and connectivity of the nodes and channels in a network. According to the functions of network nodes in a network topology, networks can be classified into direct and indirect networks. In a direct network, each node is both a terminal and a switch node. In a mesh topology, each node is used as a terminal node

connecting with a functional host and as a router node switching data to their destinations.

Many NoC designs apply mesh topology since its simple structure and the ease of placement. Another topology of direct networks which has been applied in NoC designs is the octagon topology. It is an eight-node ring network with extra links between each pair of opposite nodes in the ring structure. In an indirect network, each node works either as a terminal or a switch. It cannot carry out both functions.

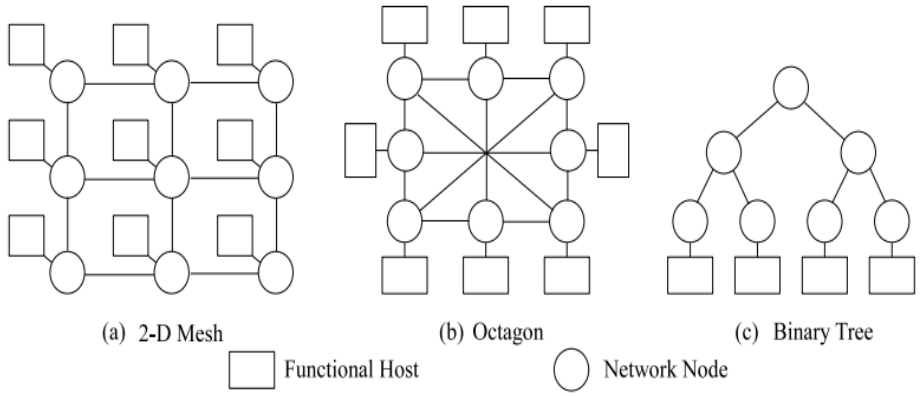

Fig 2.4 Network Topology Examples

Besides network topology, the routing method is another issue that needs to be considered in the network layer of a NoC design. After a network topology is set, a routing method is used to decide the path that data will be transferred from the source node to the destination node.

Depending on where the routing decision is made, we can have source routing and distributed routing. By source routing method, the entire path of data transfers is determined by the source node before data transfers. By distributed routing, each router node decides the next node where the received data will be sent.

Depending on the information on which the routing decision bases, routing methods can be classified into deterministic routing and adaptive routing. Deterministic routing means that the data transfer path is determined only according to the source and destination addresses. Whereas, with adaptive routing method, the path is deduced not only by the source and destination information, but also by the dynamic network conditions, such as traffic congestion information in the network.

Depending on the length of the decided path, minimal routing and non-minimal routing methods can be differentiated. If a selected path is one of the shortest paths between the source and the destination, this method is called minimal routing. Otherwise, it is called non-minimal routing method.

A routing method applied in NoC designs can be a mixture of different routing categories. In X-Y routing, the data are transferred along the rows first, then are moved along the columns toward the destination in a 2-D mesh network. Because adaptive routing involves dynamic arbitration mechanisms which incur complex node implementation, deterministic routing is normally applied in NoC designs.

2.3.4 Transport Layer

Transport layer protocols establish and maintain end-to-end communication between transport level entities. The concerned design issues in this layer include control and Quality of Service (QoS) management.

Flow Control is the mechanism that determines the allocation of resources for data as they progress along their routes. According to the way of utilizing the channels between network nodes, two different approaches, circuit-switching and packet-switching can be applied. In a circuit-switched network, a dedicated path from source to destination is set up before data transport and reserved until the transport is complete. In a packet-switched network, the data are transferred in form of packets. There are no channels set up for a

data packet. All packets travel to their destinations by sharing the existing channels among nodes and following their paths determined by a routing method. The main disadvantage of circuit-switched networks is the lower efficiency of channel usage than the packet-switched network, which is caused by setting up dedicated paths for data transport. Therefore, packet-switching method is popular in NoC designs.

Normally, an on-chip network needs to include buffers to facilitate data transfers. According to the granularity at which buffers and channels are allocated and the way of forwarding data along their routes, flow control methods can be classified into packet-buffer flow control and flit-buffer flow control. A flit is the minimum unit in a packet that can be recognized by a flow control method.

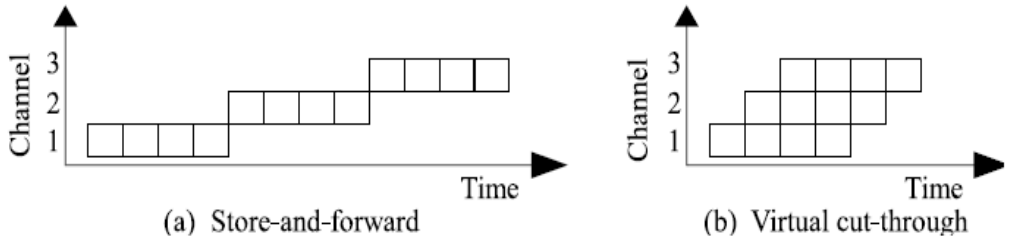

Fig. 2.5 Packet-Buffer Flow Control Method

Two basic packet-buffer flow control methods are store-and-forward and virtual cut-through. With store-and-forward method, a packet will not be forwarded to the next node along its path until all flits of the packet are received by the current intermediate node. Therefore, the disadvantage of this method is the high packet transfer latency caused by inefficient usage of channels. Hence, virtual cut through method is proposed to solve this problem by immediately forwarding the received packet flit to the next node if the buffer and channel resources are available for the whole packet, without waiting for the entire packet to be received. However, there are two main shortcomings of virtual cut-through, or of any other packet-based flow control methods. One of them is the inefficient usage of buffers caused by allocating buffers in units of packets. Another shortcoming is that

the contention latency is increased by allocating channels in units of packets. The blocked packet needs to wait for the whole packet in transmission passing through the channel before it can acquire the channel. These shortcomings can be overcome by allocating resources in units of flits rather than packets.

A popular flit-buffer flow control method is wormhole method which operates like virtual cut-through, but with resources allocated to flits rather than packets. It means that a flit only needs to acquire one flit buffer and one flit channel bandwidth before it can travel to the next node, which relieves the requirement of resources in comparison with virtual cut-through method. Whereas, with wormhole method, a packet in transfer occupies multiple channels when its flits are traversing along the channels one by one. This will cause a problem if the current packet is blocked during transfer. In this situation, virtual-channel method is proposed to solve this blocking problem by associating multiple buffers to one physical channel.

Generally, the flit-buffer flow control methods are preferred in NoC designs because of its efficient usage of buffers and channel bandwidth.

Another issue concerned in the transport layer of NoC design is QoS. It refers to the service qualification that is provided by the network to its users. The QoS of the NoC designs is classified into two basic classes, Best-Effort (BE) services ad Guaranteed Services (GS). In BE services, the network makes no strong guarantee about the delay or loss, while the GS scheme can guarantee a certain level of performance as long as the injected traffic complies with a set of restrictions. Both type of QoS have been applied in NoC designs. Because GS service demands more resource reservation and complex control logic than the BE service does, it is more expensive to support GS service in a NoC design.

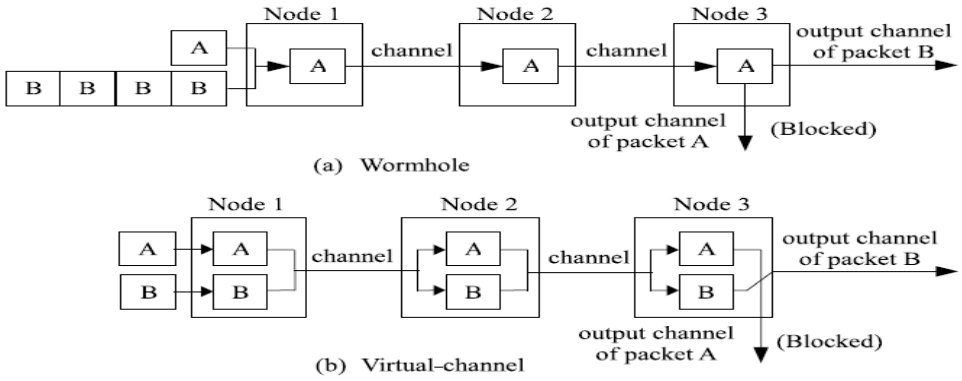

Fig 2.6 Flit-Buffer Flow Control Methods

2.3.5 Session/Presentation/Application Layer

These three layers handle the communication processes of an interconnection network in high levels. Session layer mainly focuses on the connections between hosts. Presentation layer concerns the data representation and security issues. Application layer supplies services to user-defined application processes using interconnection networks. Generally, the services and functions of these three layers will be implemented by processors or software. Therefore, a NoC design normally does not need to directly handle the issues related to these layers.

The OSI model is only a reference for designing interconnect network. Therefore, it only gives a guideline for designing an on-chip network rather than a regulation. From the above discussions about the OSI model and NoC design issues, we can see that the NoC design issues are generally within the three or four lowest layers in the OSI model and the boundaries between the design issues according to the layer definitions are not very strict. The presented design issue in this subsection do not mean a complete list of all possible design issues of on-chip network, or rather, they are some typical NoC design issues addressed according to the OSI model.

2.4 Examples of existing NoC Designs

A lot of research work about NoC structures has been carried out with different application requirements and backgrounds. There is no standard way to classify or summarize them. In this section, some examples are introduced to present the diversity of the existing NoC designs.

2.4.1 Different Topologies

As presented in section 2.3, 2-D mesh is the most widely applied topology since its simple structure and tidiness for placement. The SoCBUS, HERMES NoC are the examples of 2-D mesh network. Based on 2-D mesh, another topology called 2-D torus can be formed by connecting each row and column of nodes in a 2-D mesh network into a ring. The torus network consists of four nodes and it is implemented on an FPGA device. Another type of topology quite different from the mesh and torus is an octagon topology illustrated in Fig.2.4(b). In the octagon NoC, the channels between every node are bidirectional links.

Besides the topologies of direct networks, indirect network topology is also applied in NoC designs. For example, SPIN is a NoC design which applies a fat-tree topology consisted of two levels of routers, four routers in each level. Each router in the first level connects with four functional hosts. Each channel is comprised of two one-way 32-bit data paths. The fat-tree topology network is further explored by a NoC design called XGFT. The XGFT NoC applies an extended generalized fat-tree topology to achieve better scalability and performance in comparison with a fat-tree network.

2.4.2 Different Data Switching Methods

The PROPHID architecture is an early developed NoC which applies circuit-switching scheme. PROPHID uses a three-stage switch structure which consists of time-division switch and space-division switch to carry out data transfers in a multiprocessor system. Because the circuit-switching scheme has the disadvantage of non-scalability and insufficient parallelism for future on-chip systems, packet-switching scheme is most widely applied in current NoC designs, such as the mentioned HERMES network, SPIN network, and XGFT network.

However, there exists switching methods which combine the characteristics of both circuit switching and packet switching in NoC designs. For example, the SoCBUS applies a Packet Connected Circuit (PCC) method which hybrids circuit switching with packet switching to transfer data in the network. It uses packet switching to set up the connection between network nodes and lock the setup as a circuit for data transmission. The Æthereal NoC developed by Philips research laboratories applies a pipelined time-division multiplexed circuit switching scheme in a packet-switched network in order to acquire contention-free routing.

2.4.3 Different Routing Methods

For the sake of simplicity, deterministic routing methods are applied in the most of NoC designs. For example, the X-Y routing has been applied in the 2-D mesh HERMES NoC, SoCIN NoC. In SoCBUS, each node makes the routing decision based on the destination address and the static knowledge of the general direction to each destination. In Octagon NoC, a deterministic minimal routing method is realized by choosing the output direction at each node according to predefined rules.

Three partially adaptive routing methods, west-first, north-last, and negative-first, are proposed for 2-D mesh networks. The common idea of those methods is that a

deterministic route is followed when certain limits are obeyed, otherwise, the routing decision made by a node can be adaptive according to traffic conditions. For example, with the west-first routing, a node always tries to transfer packets firstly to the west direction of the source node whenever it is possible, otherwise, routing direction is adaptive to the traffic condition. The conclusion is that X-Y routing appears as the better choice in most situations in HERMES NoC.

2.4.4 Different Flow Control Methods

Wormhole and virtual-channel methods are two most frequently used flow control methods in NoC designs. Because the wormhole method requires less buffers and simpler control, it is easier to be applied in NoC designs. The widely accepted Æthereal NoC applies wormhole method in its best-effort router. HERMES, SoCIN, and SPIN also apply wormhole method. The virtual-channel applies 10K bits storage for virtual channels at each input controller. Virtual channels are also applied in a router design to support different QoS.

2.4.5 Different QoS Strategies

As addressed in section 2.1, GS and BE are two types of QoS applied in NoC designs. GS provides predictability of data transfers, while BE service has higher resource utilization. NOSTRUM is an example of NoC design which provides GS. It uses looped containers implemented by virtual circuits to support GS in a mesh network. While in Æthereal NoC, both GS and BE services are provided by using a combined GS-BE router structure. The router includes two parts; one part applies pipelined circuit switching to implement its guaranteed service, while the other part applies input-queued wormhole flow control to provide best-effort service. The design provides differentiated QoS between GS and BE by allowing higher priority data streams to overtake those of lower priority in virtual channels.

2.4.6 Different Implementation Strategies

As the design requirements for a NoC may vary largely depending on the applications, there is no a universal design which suits for all applications. Therefore, some NoC designs, e.g. SoCIN and Xpipes, realize the network components in a soft format which can be customized for a specific application. With the support of specialized design tools, e.g. XpipesCompiler, many design parameters, such as topology, network interfaces, and switch structures, can be customized to meet the requirements of a specific application during the design stage. Of course, the changeable design parameters in this type of NoC design can not be arbitrary. For instance, the topologies supported by SoCIN NoC only include mesh and torus. However, these choice limitations are reasonable since it is impossible to predict and meet all possible application requirements in one design.

2.4.7 Different Communication Synchronization Strategies

As addressed in section 2.1, GALS communication scheme is introduced in NoC to deal with the issue of multiple clock domain data transfer. Thus, asynchronous circuit design is applied in some NoC designs to implement GALS scheme. A NoC design called CHAIN is such an example of an asynchronous NoC. It applies self-timed logic to build pipelines, multiplexing structures, and steering latches to transfer data with handshake protocols. Another GALS NoC example is MANGO which applies OCP compliant network adapter block to connect the functional blocks with its asynchronous communication network. It also provides both guaranteed and best-effort services by utilizing virtual channels. Nexus NoC is an example of GALS NoC different from router-based NoC. It applies a 16-port asynchronous crossbar structure to build an on-chip data switch.

CHAPTER 3

3.1 Understanding GALS

Globally Asynchronous Locally Synchronous systems are an intermediate style of design between these two. GALS systems contain several independent synchronous blocks which operate with their own local clocks and communicate asynchronously with each other. The main feature of these systems is the absence of a global timing reference and the use of several distinct local clocks (or clock domains), possibly running at different frequencies.

The idea of GALS system design is in itself not new. Interest in GALS design is now growing due to the following reasons:

- **Global clock distribution:**

  Trends of increasing die sizes and rising transistor counts may soon lead to a situation in which distributing a high-frequency global clock signal with low skew throughout a large die is prohibitively expensive in terms of design effort, die area, and power dissipation. GALS systems eliminate the need for careful design and fine-tuning of a global clock distribution network.

- **Design reuse:**

  Designers are now seriously exploring opportunities for reusing IP cores, and system-on-chip design is gaining popularity. Integrating several cores on one chip may not always be possible with a single clock system; different cores may have different clock requirements and operating frequencies. GALS systems with standardized asynchronous interfaces will facilitate design reuse.

- **Inertia:**

  While a fully asynchronous design style promises to solve both the above problems, a complete migration from synchronous to asynchronous systems is not

likely to happen in the immediate future; CAD tools for asynchronous design are mature, but not commercially strong yet.

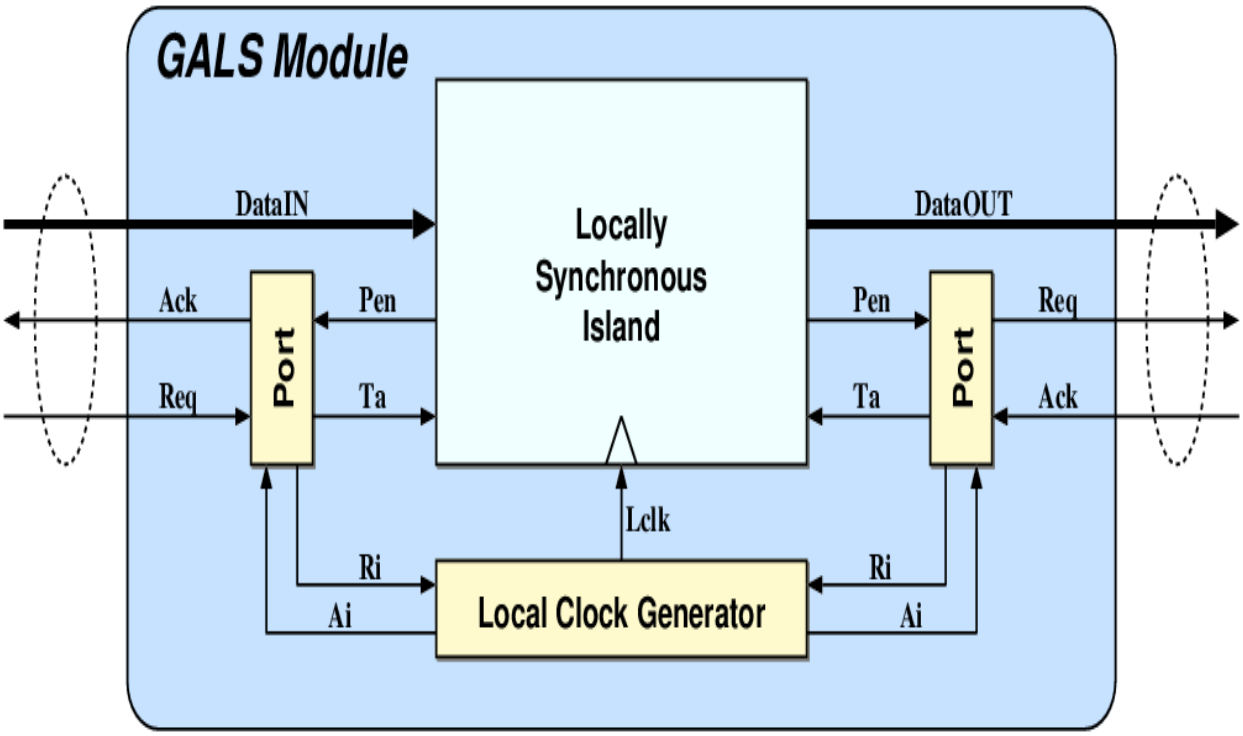

Fig 3.1 A GALS Module

The port, once activated by *Pen* (A), immediately activates the *Req* signal (B) and waits for *Ack+*(C). The local clock is only paused after the *Ack* signal is received. After the clock is paused (D), the data can be safely sampled. At this point, the data transfer is effectively concluded and the *Ta* signal is activated (E). Afterwards, the handshake signals are returned to their idle states and the clock pause request signal *Ri* is

deactivated (F). The *Ta* signal remains active (G) as long as the *Pen* signal remains active (H).

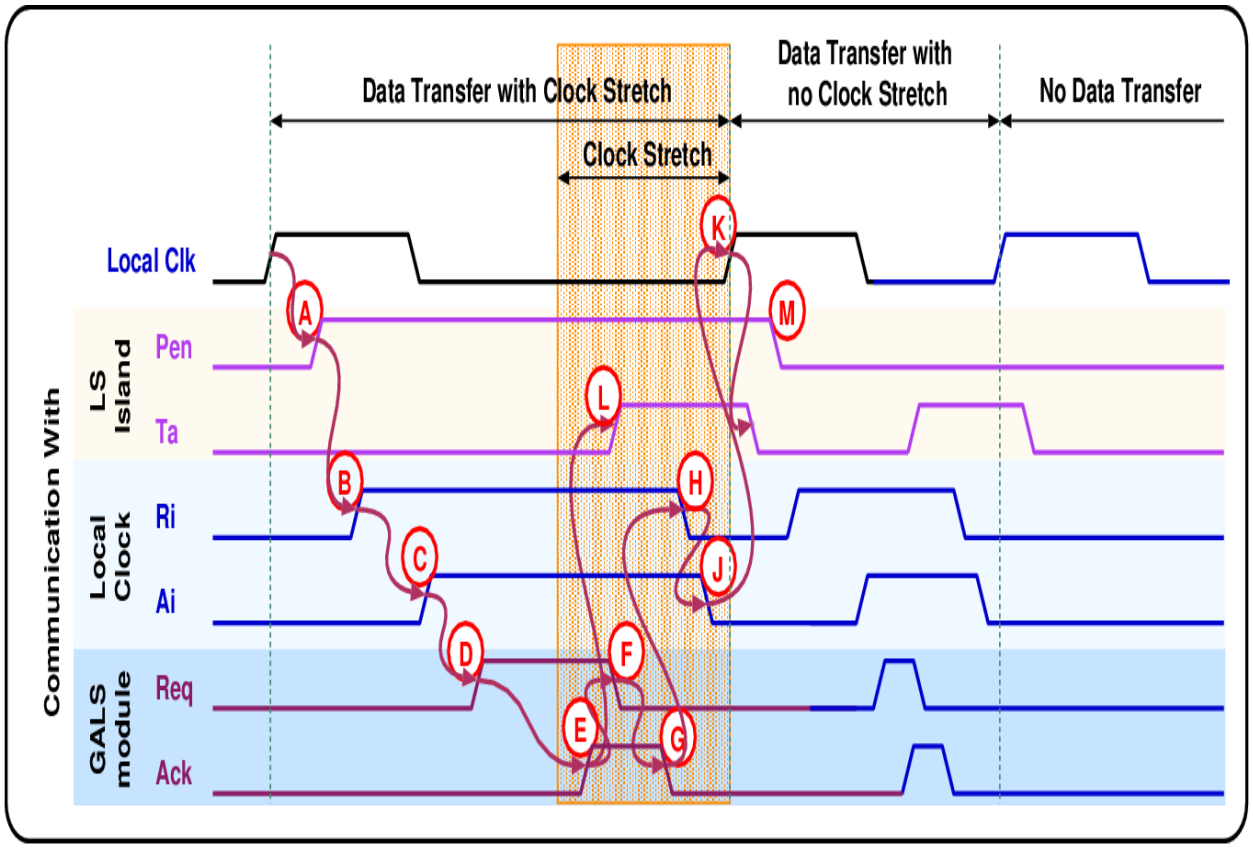

Fig 3.2 Timing Diagram

3.2 Different GALS Arcitectures

3.2.1 Handshake based GALS Architecture

In the handshake-based GALS architecture, the synchronous components communicate directly via handshaking schemes. A receiver-transmitter unit (RTU) is added to each component to ensure proper execution of the request-acknowledge based handshake protocol. Each signal (carrying valid data) is augmented with two extra signals for control purposes: request and acknowledge.

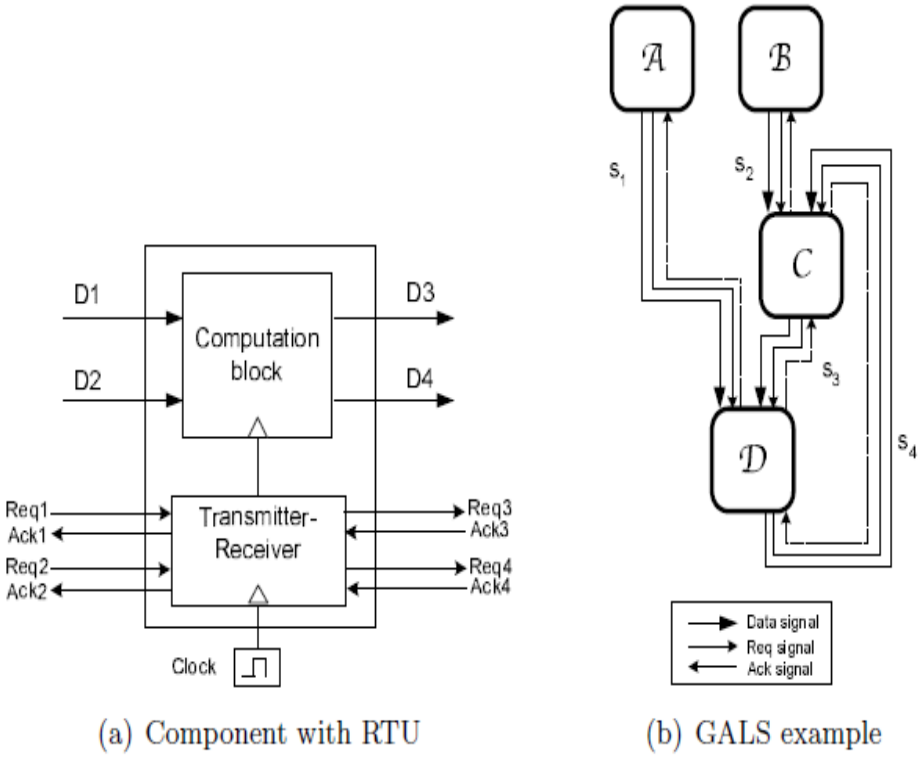

Fig 3.3 Handshake based GALS Architecture

In a network, a synchronous component executes when the following conditions hold: (i) all its input request signals are requesting (req=1), (ii) all its input acknowledge signals are waiting for new request (ack=0). Once, both these conditions hold, the component executes based on its clock. Until these conditions are true, the synchronous component is disabled.

3.2.2 FIFO based GALS Architecture

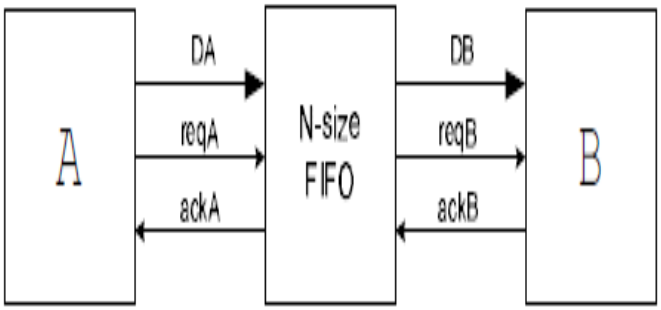

Fig 3.4 Asynchronous FIFO with Handshake

The asynchronous FIFO placed in between two components (A and B) will require RTUs on both its ends. Component A's RTU will communicate with RTU of the FIFO facing towards A. The RTU of the FIFO facing B will communicate with the RTU of B. For the four-phase handshake protocol, four handshakes will be required to communicate a single data from component A to FIFO, and the same from the FIFO to component B. In other words, a total of 8 handshakes will be needed to communicate a data from component A to component B. In the case of two- phase handshake, the total handshakes for exchanging one data will 4. Such an architecture will be very expensive with respect to the performance of the design.

3.2.3 Controller based GALS Architecture

The Local Control Units (LCUs) of the components communicate asynchronously with a central control unit (CCU) to request for a permission to execute.

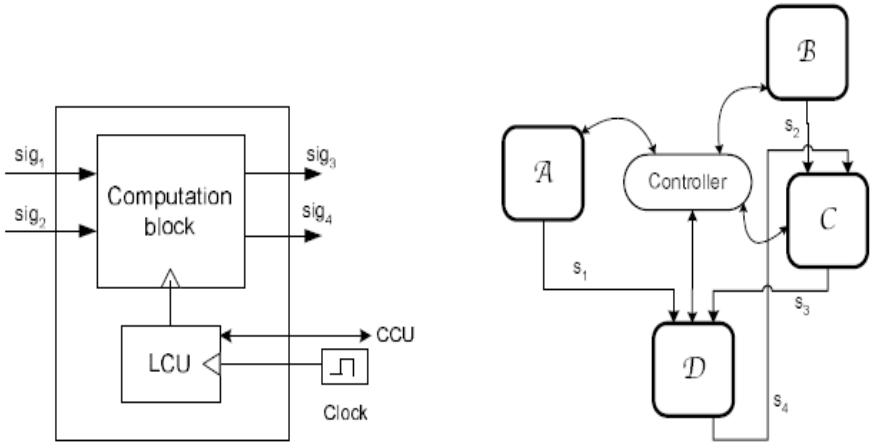

Fig. 3.5 Controller Based GALS Architecture

3.2.4 Lookup-based GALS Architecture

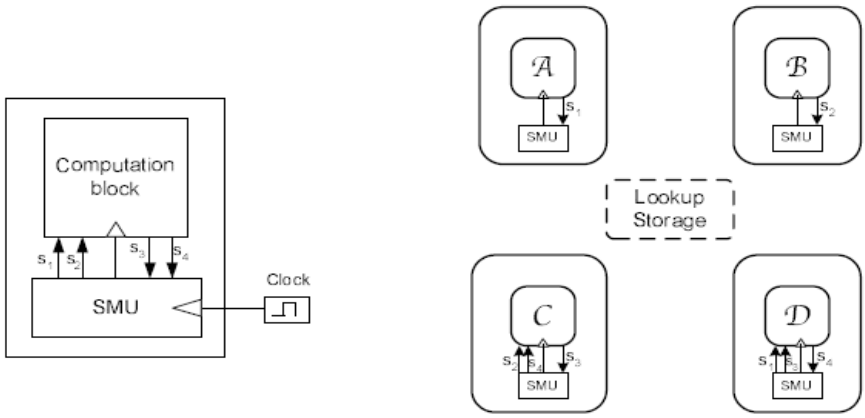

Fig 3.6 Block Diagram of a Component in Lookup- based GALS Architecture

A storage mapping unit (SMU) is added to each component in the lookup-based GALS architecture. The communication between the components is based on reading and writing from a lookup storage which is placed on the chip for fast access to data.

3.2.5 Comparison of different Architectures

The handshake-based GALS architecture should be chosen as the target architecture when there is a constraint on adding additional elements such as communication media. Here, the cost associated with additional signals such as placement and routing is not an issue. FIFO-based architecture is a good choice as the target architecture for GALS, if additional signals can be added easily with FIFOs. Such architecture would be best for performance driven applications. The controller-based GALS architecture is better if there is a constraint on number of signals can be added, and the ratio of the components in the design over the number of inter-component signals is higher. Hence, less number of signals will be added in this architecture than the handshake-based architecture. If addition of extra storage elements on the chip is not an issue, and storage accessing time is assumed to be little, then the lookup-based GALS architecture is best. It was realized by the example that the Lookup-based GALS architecture had the best performance if there are no constraints for additional elements/signals on the chip, and the accessing time was assumed to be negligible.

3.3 Applying GALS Scheme into ON-CHIP NETWORKS

As mentioned in before, the number of processing or functional components in an onchip system becomes larger and larger. Currently, a 64-core on-chip system, has already been produced. It is believed that a future on-chip system will consist of sea-of processors in one chip. Besides the growing number of system components, the functionalities of system components also become largely different from each other. It means that an on-chip system may include different processors for different computation tasks, varied hardware accelerators for varied functions, and various interface controllers for various peripheral devices. Therefore, these heterogeneous system components have different optimal working clock frequencies according to the tasks that they are handling. When

integrating all the heterogeneous components into an on-chip system, coordinating different clock domains is a challenge.

3.3.1 Multi-Clock Challenge and GALS Scheme

From the viewpoint of a chip design, for large high-speed globally synchronous ASICs, designing the clock distribution net becomes a troublesome task because of the problems caused by clock skew, by the growing die sizes and shrinking clock periods. At the same time, the power consumption is increasing tremendously because the working clock frequency driven by demanding applications is getting higher in the scale of giga-hertz.

Therefore, one solution of the challenges mentioned above is to enable different processing or functional system components to work at their own clock rates. Thus, the following challenge that a SoC designer needs to handle is how to integrate the clock independent components into one system. In this situation, GALS scheme is proposed to solve the system integration challenge. The GALS scheme is firstly introduced to prevent metastability by stretching local clocks. The basic idea of applying GALS scheme into on-chip systems is to partition the system into several independently clocked domains that communicate with each other in an asynchronous fashion. The GALS scheme is the basis of the NoC structures designed and realized in this work. The method and challenges of designing a GALS on-chip network will be presented in the following two sections.

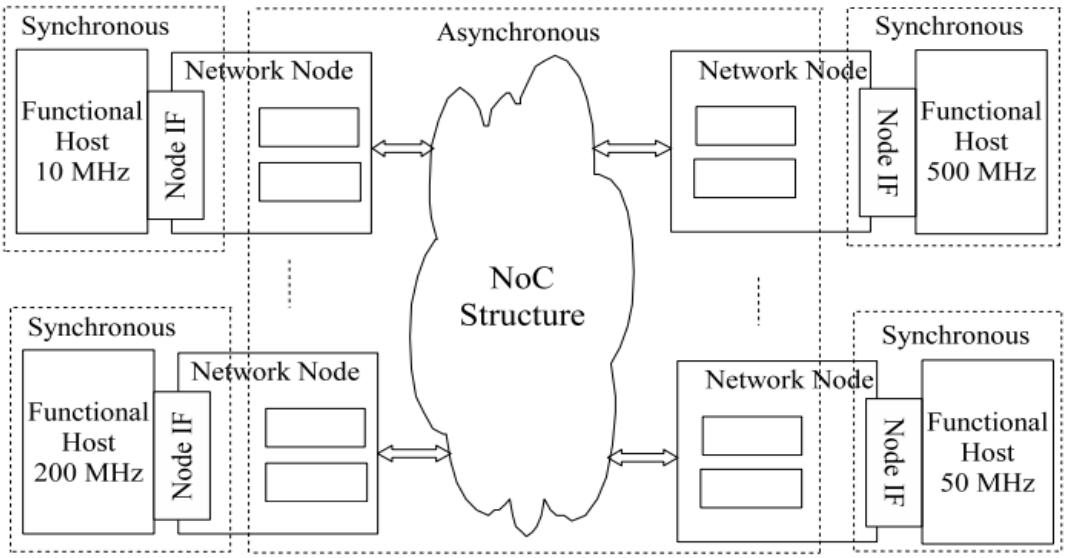

Fig 3.7 A Method of applying GALS scheme in a NoC design.

3.3.2 The Synchronization in GALS NoC

In an on-chip system, communication tasks among system components are performed by the on-chip network. Thus, the issue of realizing GALS scheme in an on-chip system equals to realize GALS scheme in the on-chip network. The method of applying GALS scheme in the NoC structures developed in this work are illustrated in Fig.3.7. From the figure, we can see that each network node contains an interface block which works at the same clock rate as the system functional block attached to it, while the blocks for global communication among network nodes apply asynchronous scheme. Therefore, the data synchronization between synchronous and asynchronous domains is the main challenge of designing a GALS NoC. The term, synchronous domain, used in this thesis refers to the group of design blocks which work under the dictation of clock signals in a SoC, while, the term of asynchronous domain refers to the group of blocks which work in a selftimed manner without any clock signals.

Many synchronization schemes or structures for data transfers among independent clock domains in a GALS system have been presented. One category of solutions is to avoid synchronization failure by adjusting the clock signal of the local synchronous module or by generating a controllable clock signal in the synchronization interface. For example, a stoppable clock structure to build a deterministic wrapper. A stretchable clock schemes to avoid synchronization failure in the interface between synchronous and asynchronous domains. A pausible clock scheme to manage the data transfers between independent clock domains without synchronization failure. An asynchronous wrapper which combines the stretchable and pausible schemes together can avoid synchronization failures caused by metastability in circuits. One common feature of those presented synchronization schemes is that they all involve specialized clock generation or control circuits which need to be implemented in circuit level. Thus, if a GALS NoC design applies one of those synchronization schemes, the whole design can not be realized in Register-Transfer Level (RTL) by using Hardware Description Language (HDL), which in turn makes the NoC design less implementation flexible and portable.

Another type of solutions of data synchronization in a GALS system is to synchronize the signals from asynchronous domain with the local clock in an arbitrary timing relationship and limit synchronization failures within an acceptable level. The most widely applied scheme in this category is the double-latching scheme. It consists of two serially connected D-Flip-Flop (D-FF) components to latch the input signals with the reference clock of the receiver. It is possible that the first D-FF enters into metastable state if input signal transitions violate the setup or hold timing requirement. In this situation, the second D-FF gives a whole clock cycle for the first D-FF to resolve the metastability before latching its output. However, in the double-latching scheme, there still exists the failure possibility if the first latch can not get rid of metastability state before the second flip-flop samples its output. Therefore, Mean Time Between Failure (MTBF) is introduced to measure the safety of a synchronizer. MTBF gives indication about how often a synchronization failure occurs. The MTBF equation consists of the time (t) allowed for synchronization, the settling time (t) of Flip-Flop (FF), the sampling clock

frequency (f_s), the frequency (f_d) of data edges which generates a metastability, and a parameter (T_w) related to the metastability window of the FF.

$$MTBF = \frac{e^{\frac{t}{\pi}}}{T_\omega \cdot f_s \cdot f_d}$$

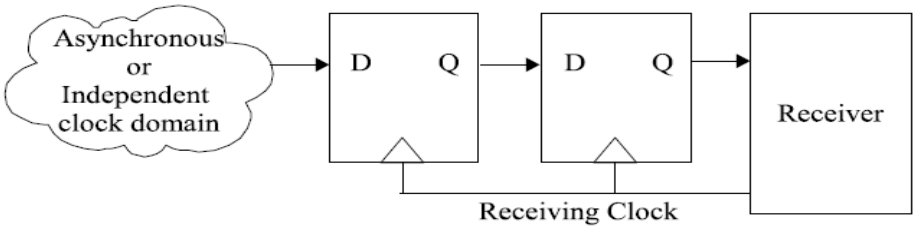

Fig 3.8 Double-Latching Synchronization Scheme.

Double-latching scheme is used for synchronizing the handshake control signals for data transfers rather than the data signals themselves. For example, when transferring data from asynchronous domain to synchronous domain, the asynchronous logic will assert a request signal after the data to be transferred are ready. Then the asserted request signal will be synchronized with the receiving clock domain through a double-latching structure as illustrated in Fig.3.8. Whereas, the acknowledge signal that the synchronous domain sends back to the asynchronous domain can be received directly. When data are transferred from synchronous domain to asynchronous domain, the double-latching scheme is only needed for the synchronous logic to receive an acknowledge signal from the asynchronous domain during a four-phase handshake process. The MTBF of most SoC designs is safe far more than enough by simply setting the resolving time window to one clock cycle. Among the published NoC designs, MANGO NoC is an example which applies the double-latching scheme to synchronize the synchronous and asynchronous domains. The designer of MANGO NoC claims that the estimated MTBF of the implemented double-latching synchronizer is longer than 8000 years. Therefore, the

simple and safe enough double-latching scheme is a reasonable choice for a GALS NoC design.

3.3.3 The Asynchronous Design for GALS NoC

In order to realize a GALS NoC design, both synchronous and asynchronous designs are needed. Synchronous design methodology and techniques have been well established and applied. Many standard design tools and design flows are developed for synchronous designs. Whereas, asynchronous design has not been widely applied after it was born in 1950s. The asynchronous designs of the GALS NoCs will be presented in this section.

Asynchronous design methods can date back to 1950s and to two people in particular: D.A. Huffman and D.E. Muller. Huffman developed an asynchronous design methodology known as fundamental-mode circuits in which the delay in all circuit elements and wires is assumed to be known, or at least bounded. The methodology developed by Muller is Speed-Independent (SI) circuits in which gate delays are assumed to be unbounded while the wire delays are negligible.

Almost all the other types of asynchronous design methods can find their roots in those two fundamental methodologies. For example, Delay-Insensitive (DI) circuit model extends the assumption of SI circuits by assuming that both gate and wire delays in circuits are unbounded. The burst-mode design methodology assumes that only the specified input bursts which can make circuits leave the current state can occur in a given circuit state, and the fundamental-mode assumption is applied between transitions among different input bursts. Ivan Sutherland developed a micropipeline structure as an asynchronous alternative of synchronous elastic pipelines. A micropipeline structure consists of a bounded-delay data path controlled by delay-insensitive control logic.

After the birth of asynchronous design in 1950s, it has not been as widely adopted as synchronous designs except several academic projects during the first several decades, such as the ILLIAC II computer developed at University of Illinois in 1960s, the first

operational data-flow computer developed at the University of Utah in 1970s, and the first fully asynchronous microprocessor developed at California Institute of Technology in 1980s. As the development of IC design in recent decades, synchronous designs face the hard challenges of clock distribution, power consumption, and design complexity. Therefore, as an alternative to synchronous design, asynchronous design gains more applications than before. For instance, Philips developed asynchronous pager chips in 1998 and a contactless smart-card chip in 2000. A series of asynchronous microprocessors called Amulet have been developed in University of Manchester from 1994 to 2000. In 2005, products based on an asynchronous NoC design were released by a company called Silistix. One common motivation of those asynchronous design applications is to utilize the advantages of asynchronous design. Several main advantages of asynchronous design are briefly introduced in the following five paragraphs as the end of this short introduction of asynchronous design.

(1) Low power consumption. Because asynchronous circuits do not need any clock signals, the power spent on clock switching in a synchronous chip is avoided. Additionally, the signal transitions in asynchronous circuits will automatically stop when there is no driven event. Therefore, asynchronous designs can achieve lower power consumption.

(2) No clock distribution and clock skew. This advantage is obvious since the lack of clock signal in asynchronous circuits. Thus, the difficulties of clock distribution and clock skew faced by synchronous designs are removed from asynchronous designs.

(3) Average-case performance. In a synchronous design, the operating speed is limited by the worst-case, called critical path, in the circuits. However, in asynchronous circuits, the operating speed is determined by actual local latencies in the circuits rather than the global worst-case latency. In most of cases, the average-case of latencies are smaller than the worstcase latency, hence, asynchronous designs can achieve better operating speed performance.

(4) Less Electromagnetic Interference (EMI) radiation. In a synchronous design, flip-flop transitions follow a certain clock frequency so that the energy spent on signal transitions concentrates within the very narrow bands around the clock frequency. Thus, the synchronized signal switching activities will produce substantial electrical noise. Whereas, the switching activities in an asynchronous circuit are correlated loosely because there is no universal timing pace, hence, they produce a more distributed noise spectrum and a lower peak noise value.

(5) Robust and adaptive. A synchronous circuit is sensitive to the delay variations caused by the variations of clock signal, supply voltage, and operating temperature related with the manufacture process and application surrounding. Whereas, because the loose timing requirement, asynchronous circuits can operate correctly under large variations caused by different manufacture processes and application environment.

CHAPTER 4

4.1 Simulation of a Mesh Network

Here simulation of mesh topology GALS module is presented. Two mesh networks are combined to make one GALS module. The links between the two mesh networks is asynchronous and the links between the nodes in one mesh are synchronous thus explaining the concept of Globally-Asynchronous and Locally-Synchronous.

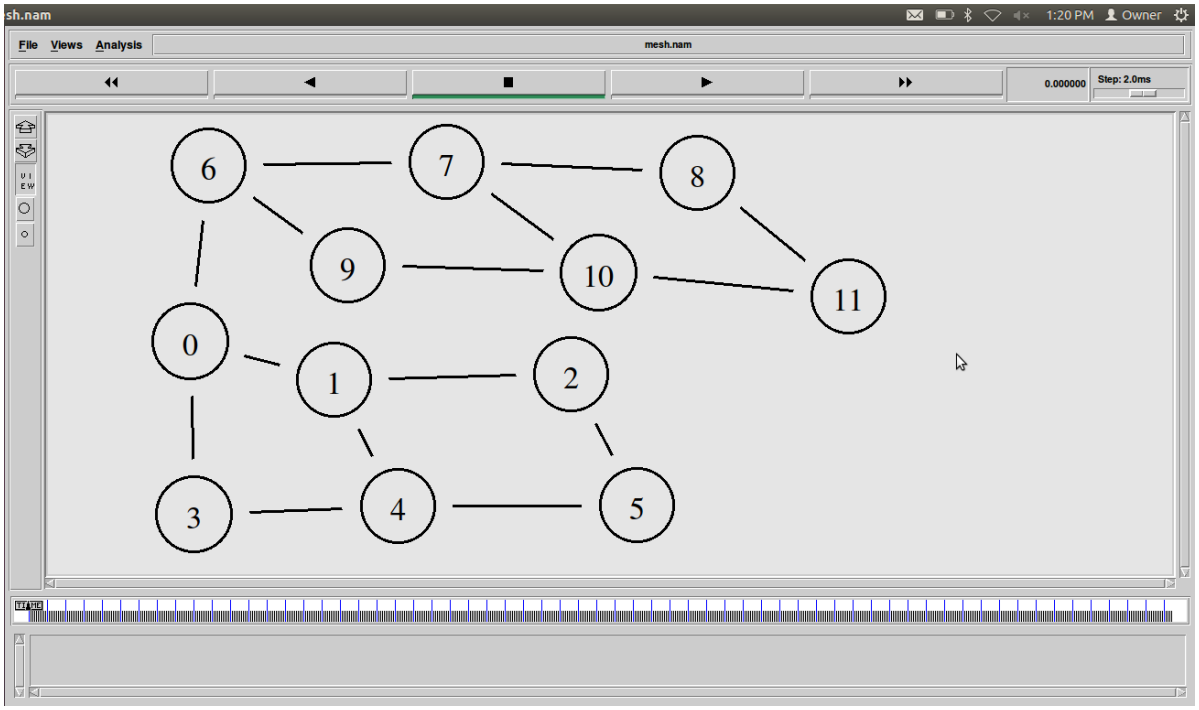

Fig. 4.1 A mesh GALS module

Packets are generated at node 0 of the first mesh network. And they are then simulated to the second mesh network.

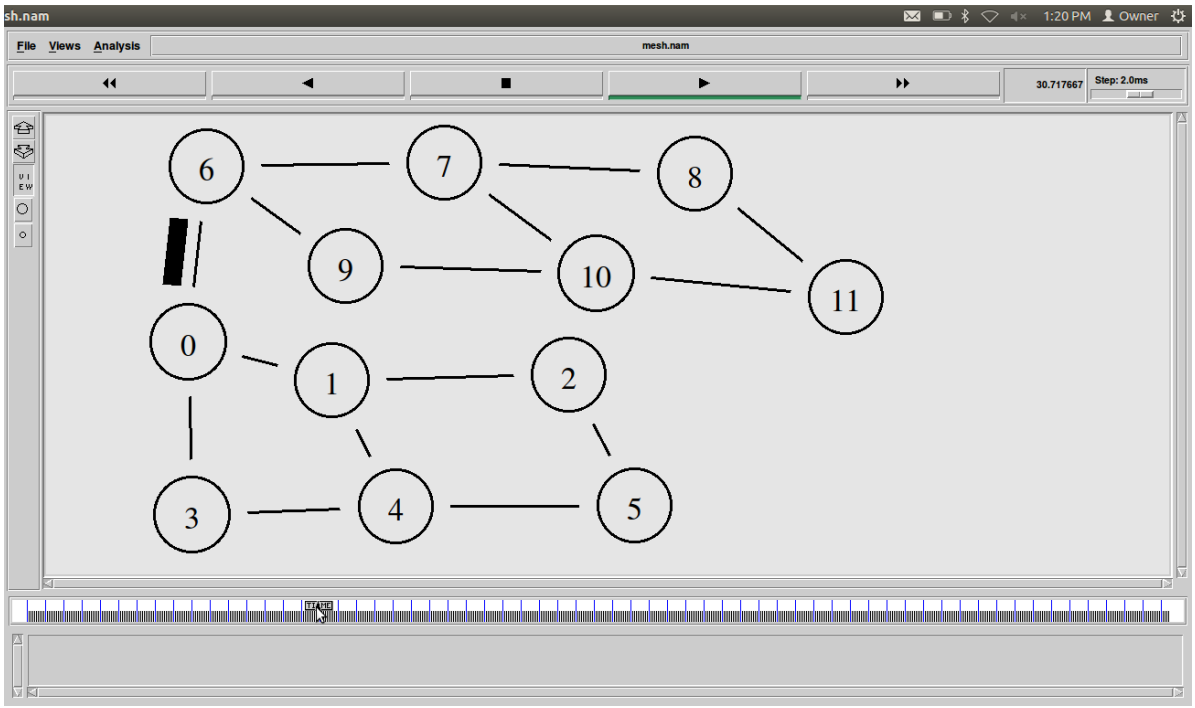

Fig 4.2 Forward Packet Simulation

When the packets reach the second network, they start moving to the destination node i.e, the sink node. The node 8 is the sink node here.

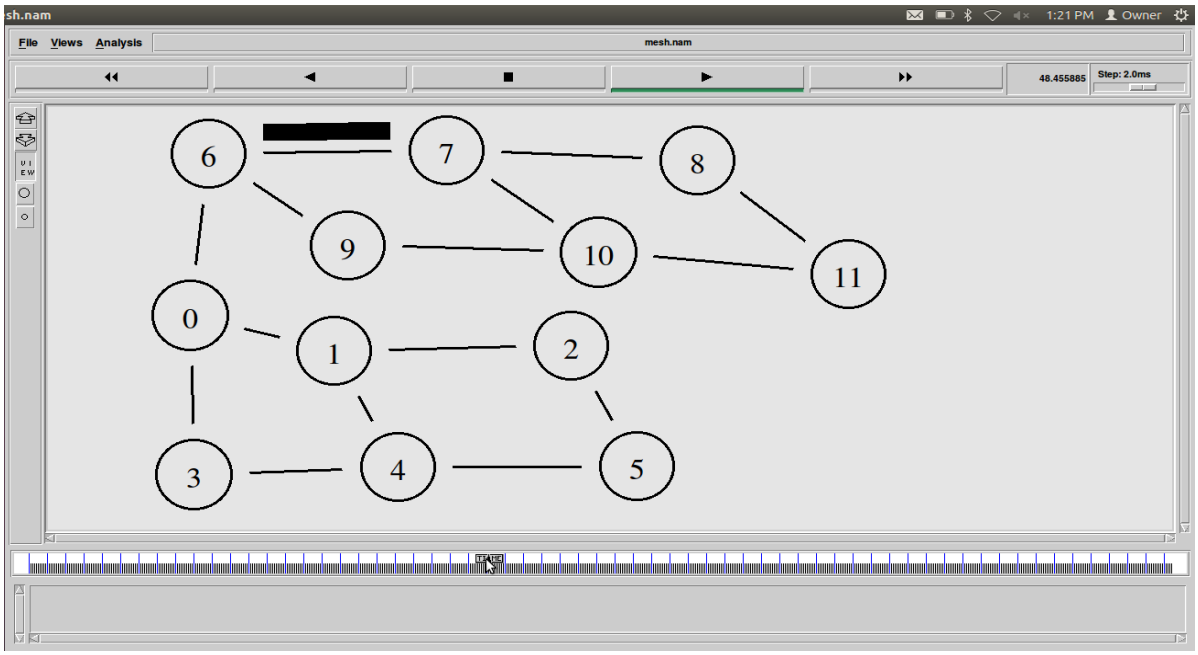

Fig 4.3 Packets in Second mesh Network

We can see in Fig. 4.3, the packets are moving in the second mesh network.

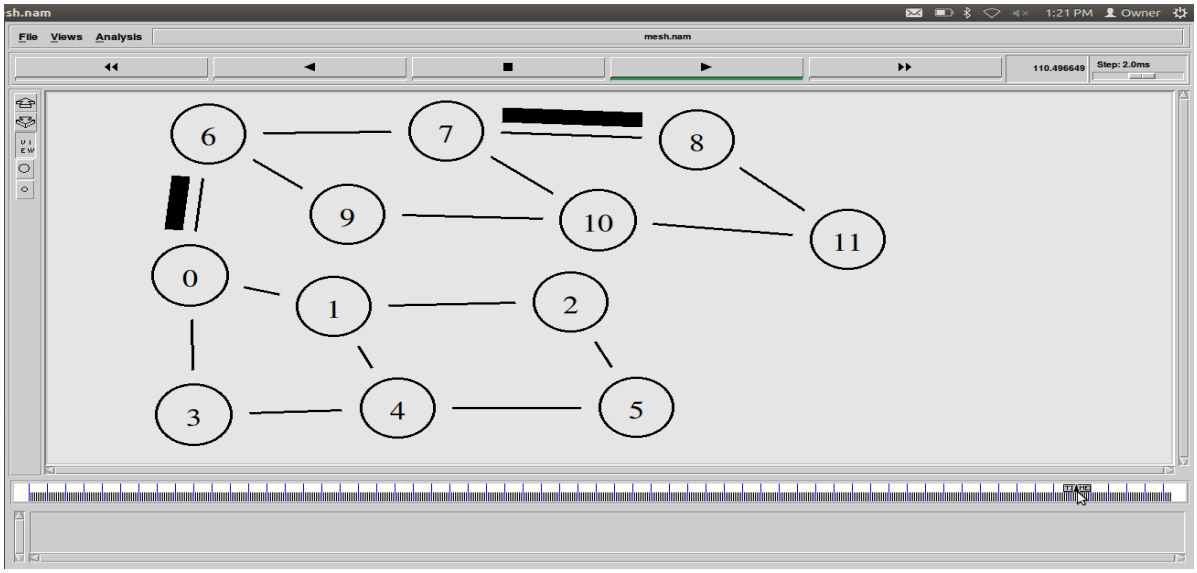

Fig 4.4 Packets reaching the destination node

4.1.1 Graphs for the Simulation

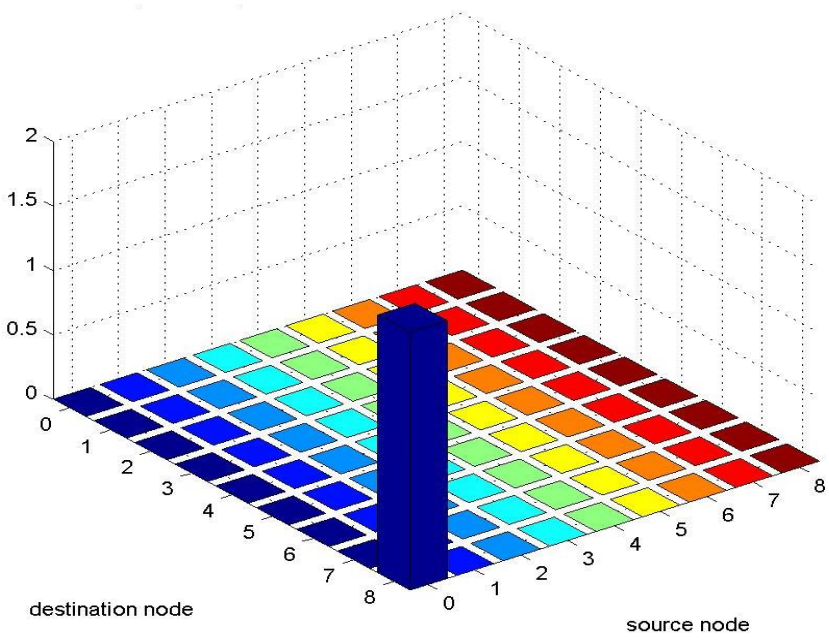

Fig 4.5 Number of packets generated packets at all nodes

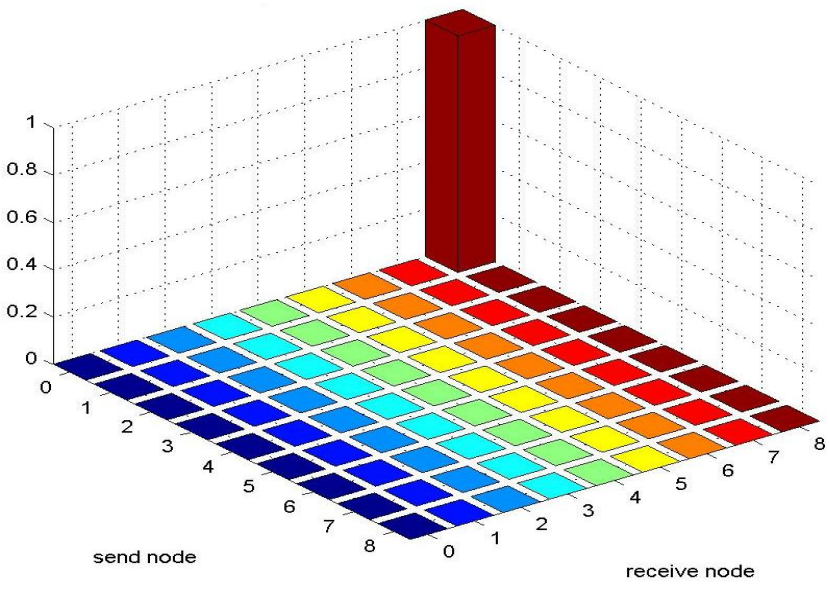

Fig 4.6 Number of received packets at all the nodes

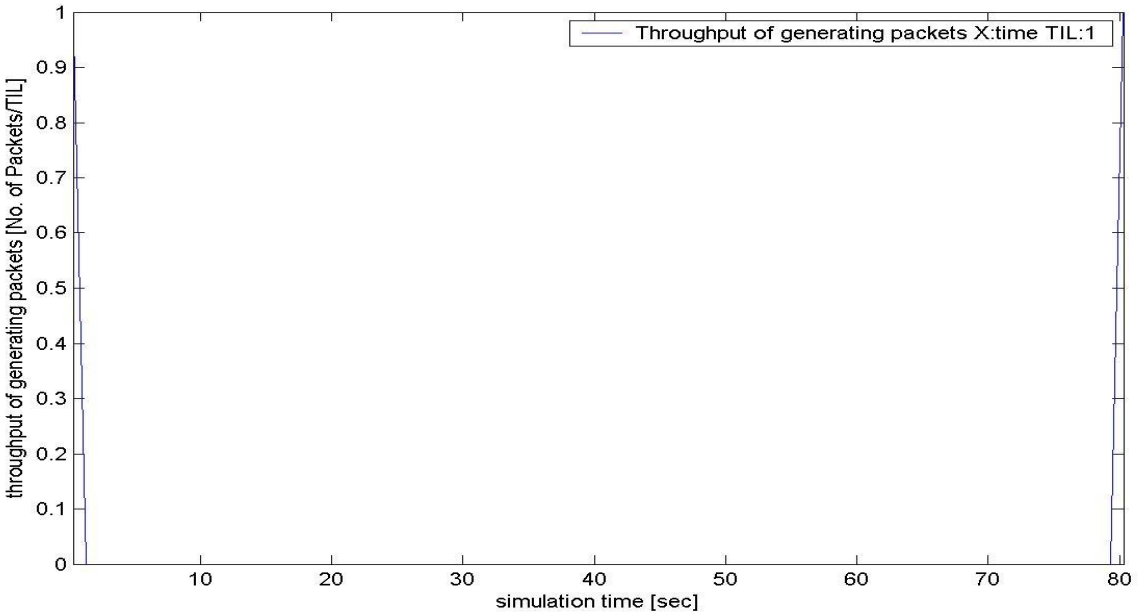

Fig 4.7 Throughput of generating packets

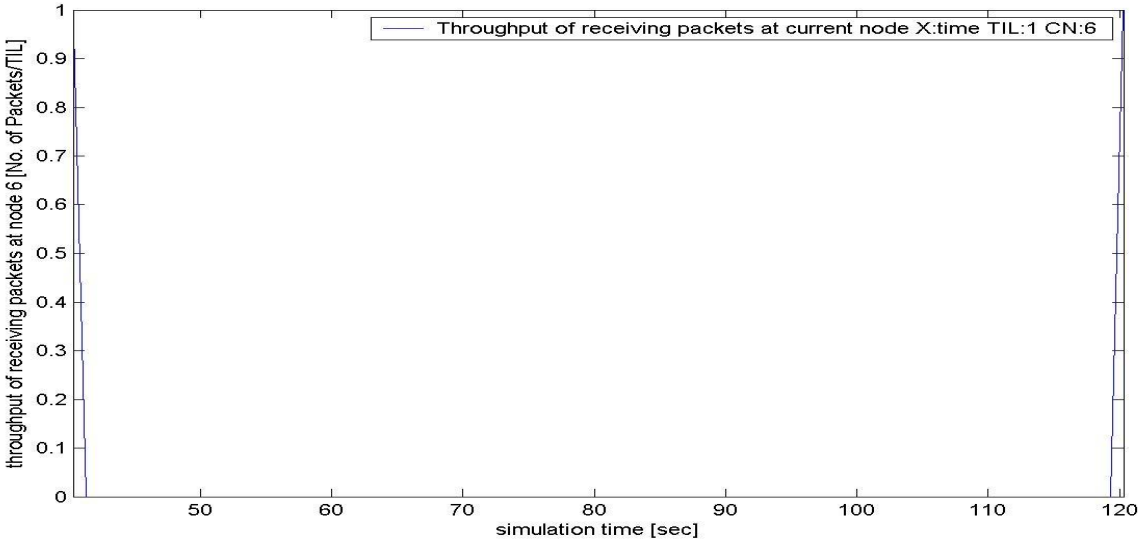

Fig 4.8 Throughput of receiving packets at node 6

4.2 Simulation of a Torus Network

Here simulation of torus topology GALS module is presented. Two torus networks are combined to make one GALS module. The link between the two torus networks is asynchronous and the links between the nodes in each torus are synchronous thus explaining the concept of Globally-Asynchronous and Locally-Synchronous.

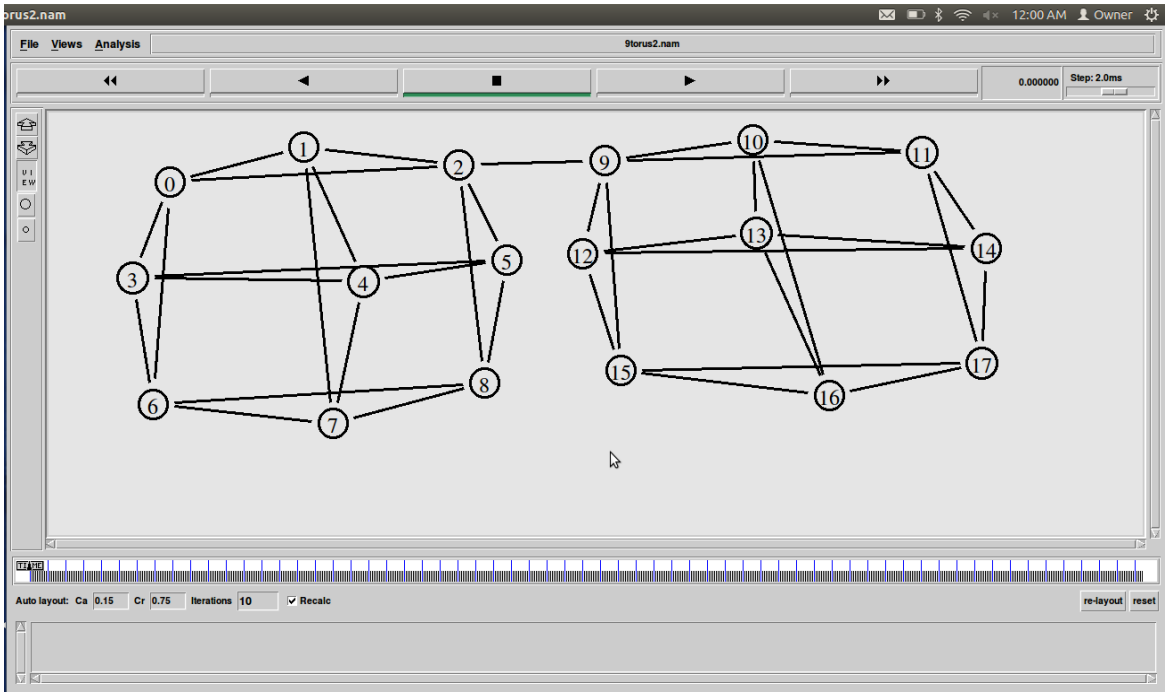

Fig. 4.9 A torus GALS module

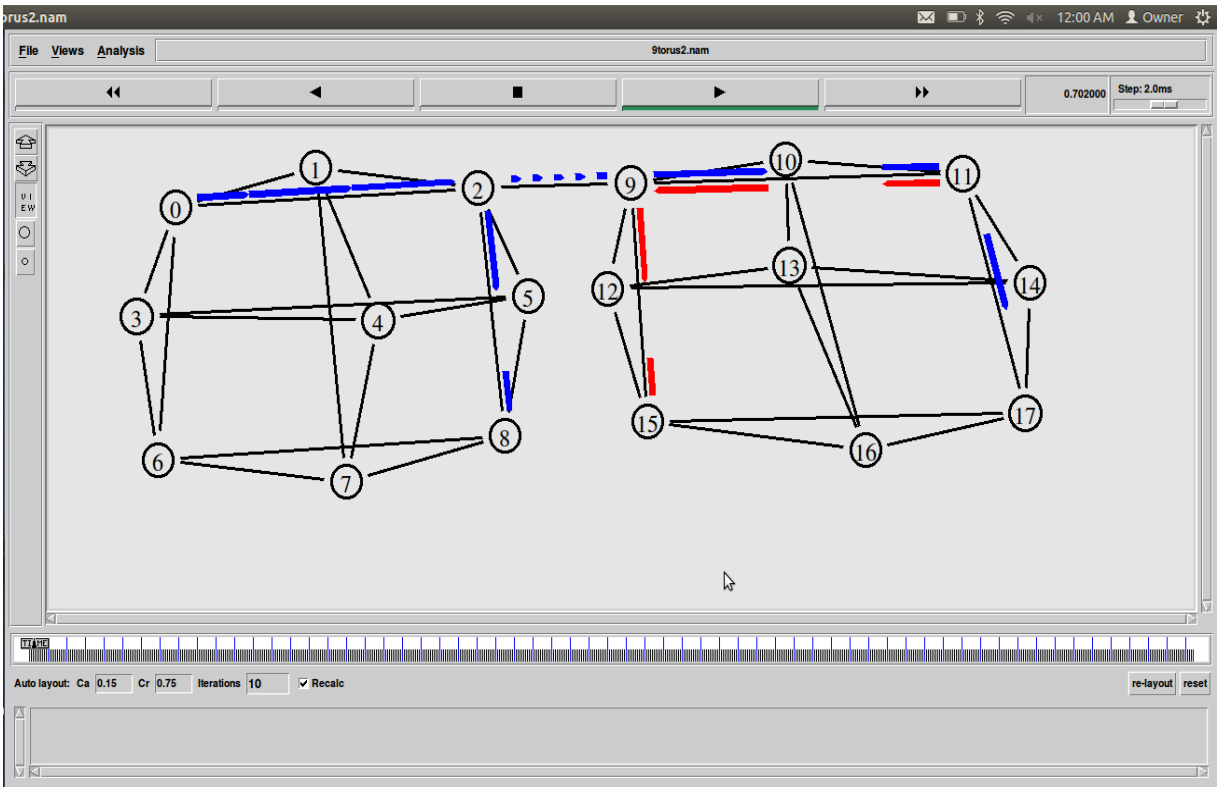

Fig. 4.10 Delay between two torus GALS module

This is a torus topology showing two modules. The link between the node 2 and node 9 is asynchronous and so the packets are reaching at different time.

4.2.1 Graphs for the Simulation

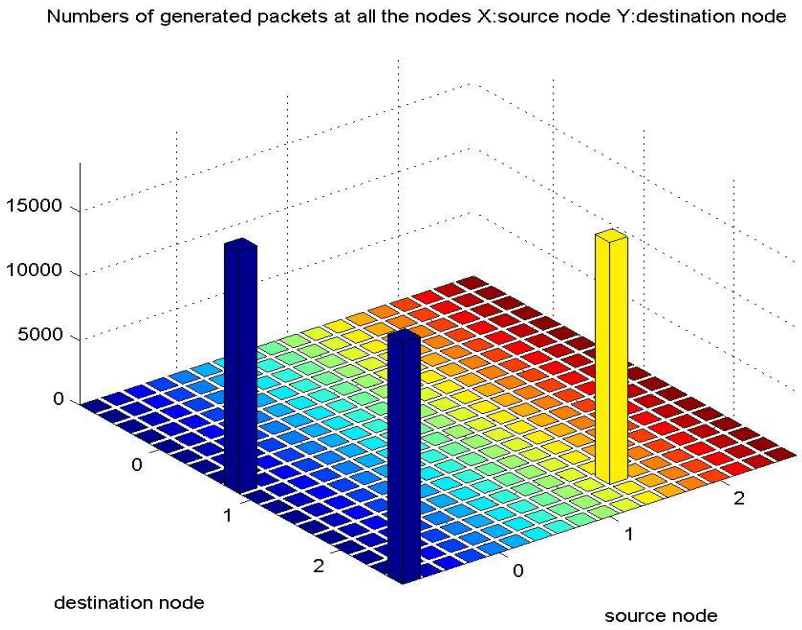

Fig. 4.11 Number of generated Packets at all nodes

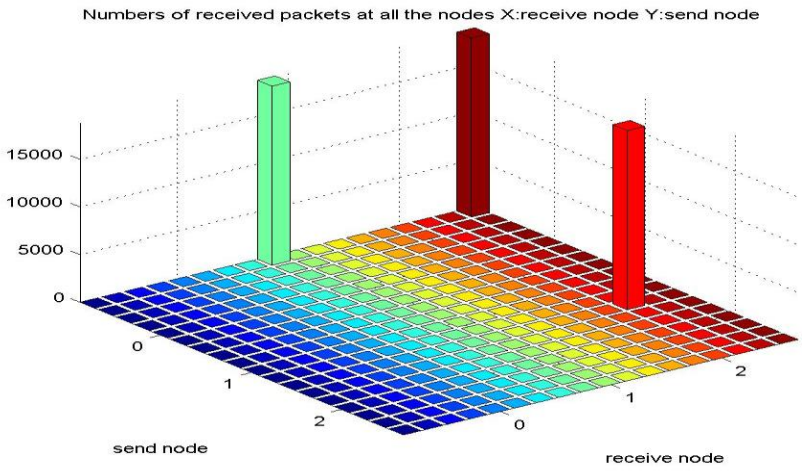

Fig. 4.12 Number of received packets at all nodes

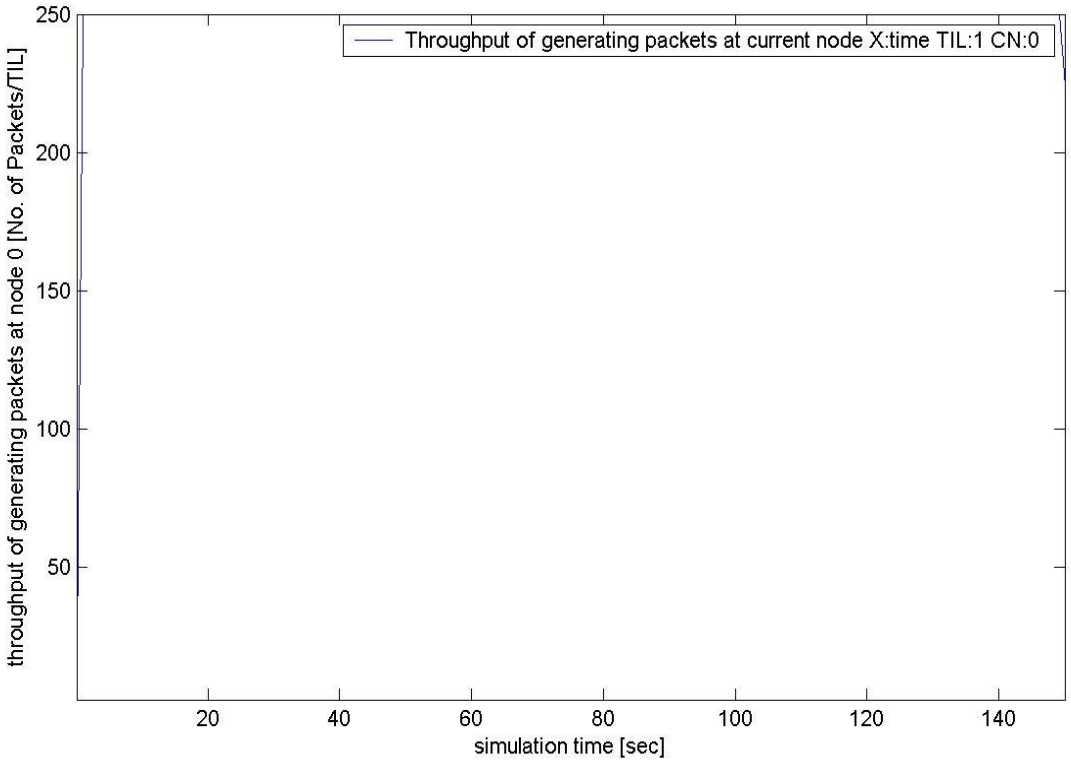

Fig. 4.13 Throughput of generating packets at node 0

4.3 Simulation of a Star Network

Here simulation of star topology GALS module is presented. Two star networks are combined to make one GALS module. The link between the two star networks is asynchronous and the links between the nodes in each star are synchronous thus explaining the concept of Globally-Asynchronous and Locally-Synchronous.

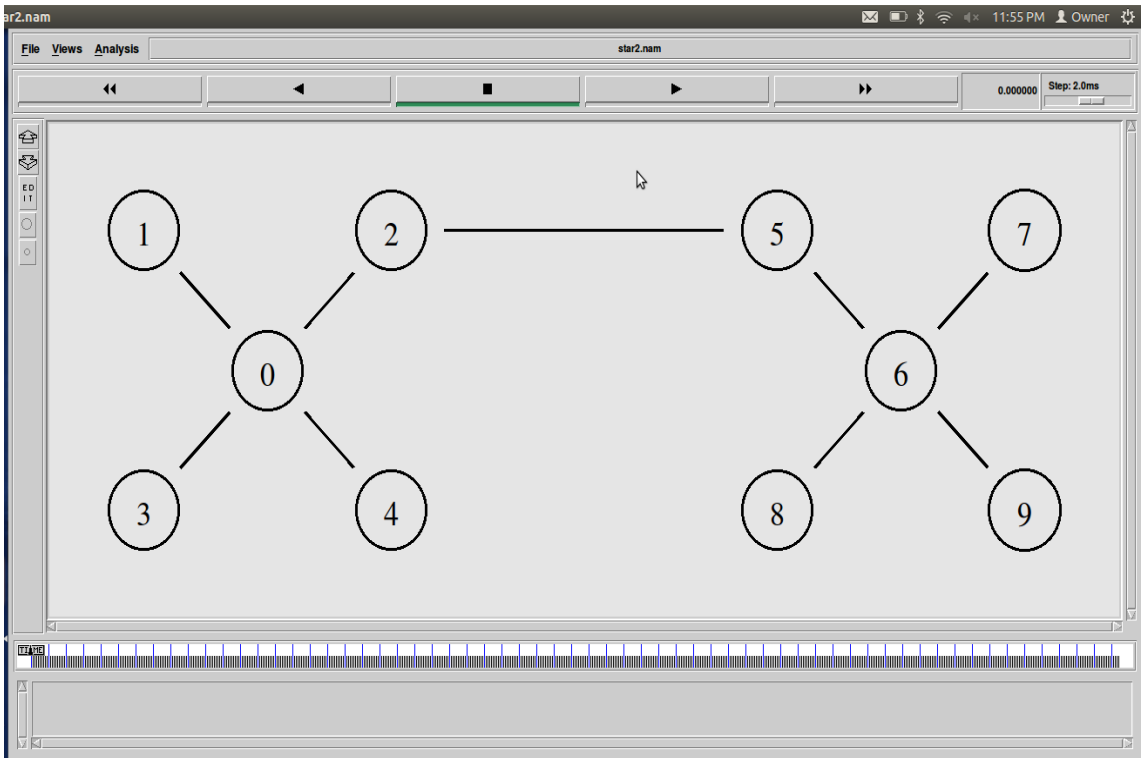

Fig. 4.14 A star network GALS module

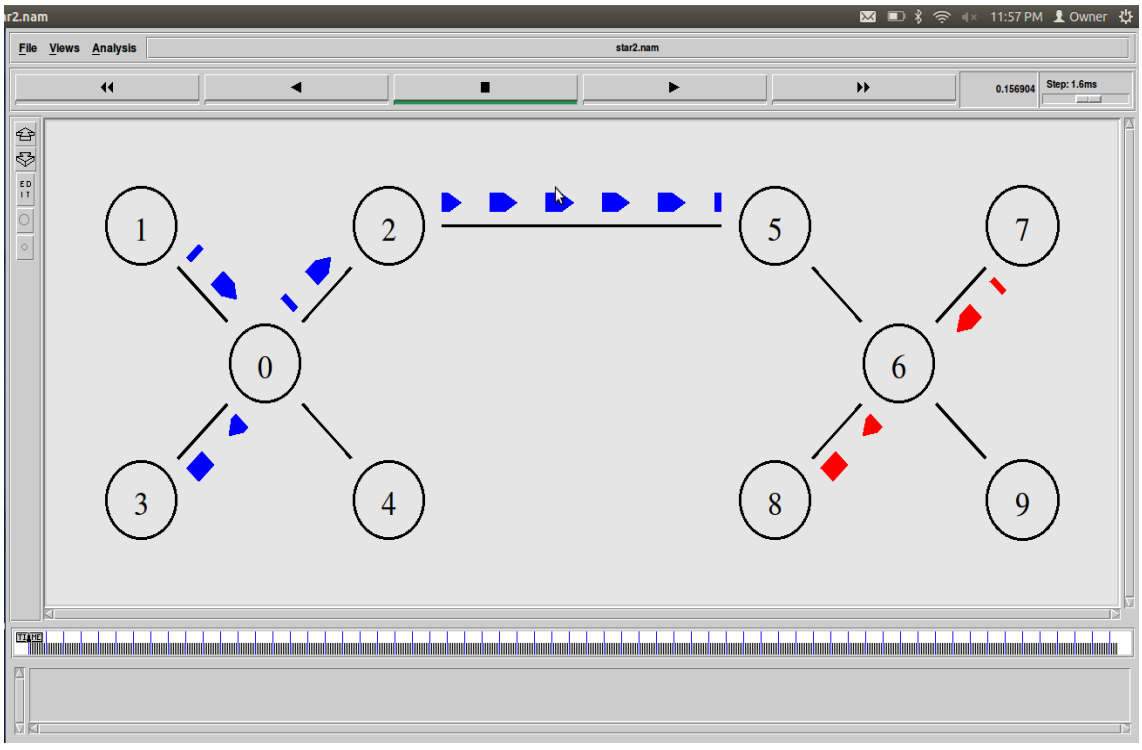

Fig. 4.15 Forward Packet Simulation

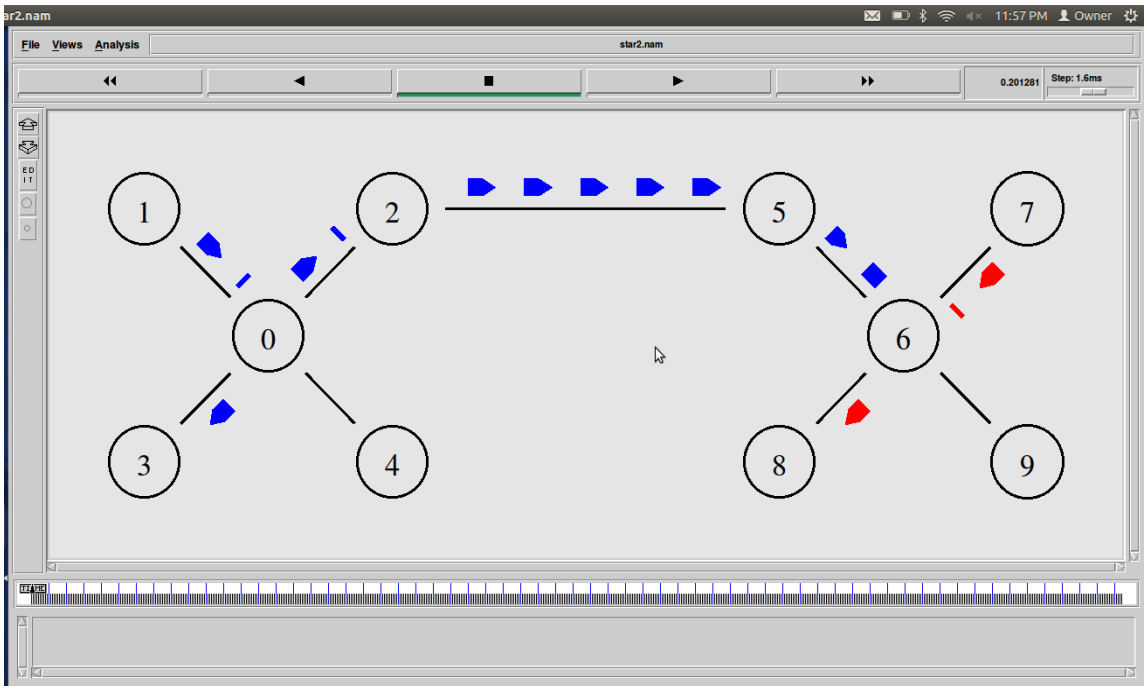

Fig. 4.16 Packets Reaching the destination Node

This is a star topology showing two modules. The link between the node 2 and node 5 is asynchronous and so the packets are reaching at different time.

4.3.2 Graphs for the Simulation of Star Network

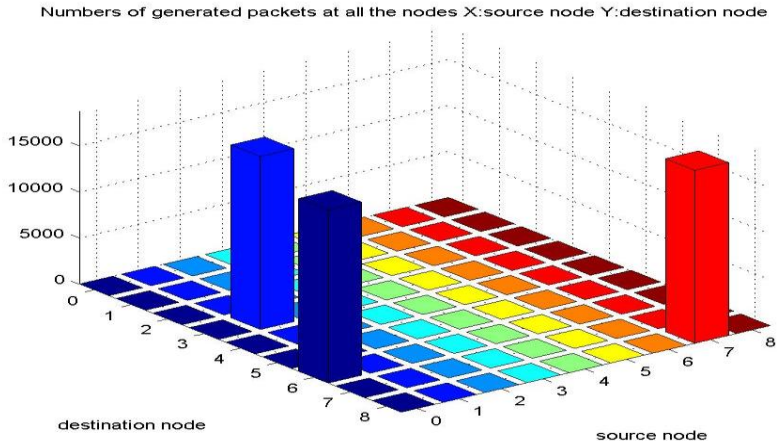

Fig. 4.17 Number of packets generated at all nodes

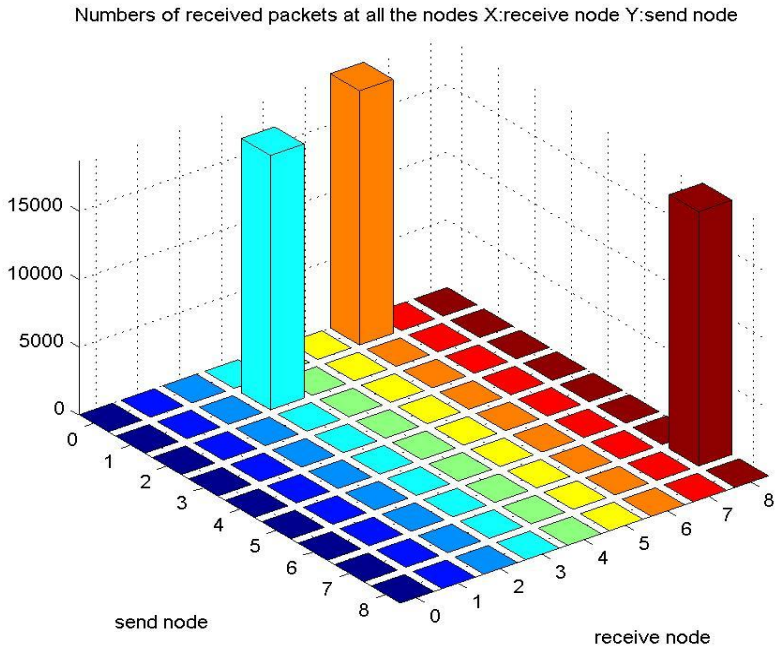

Fig. 4.18 Number of received packets at all nodes

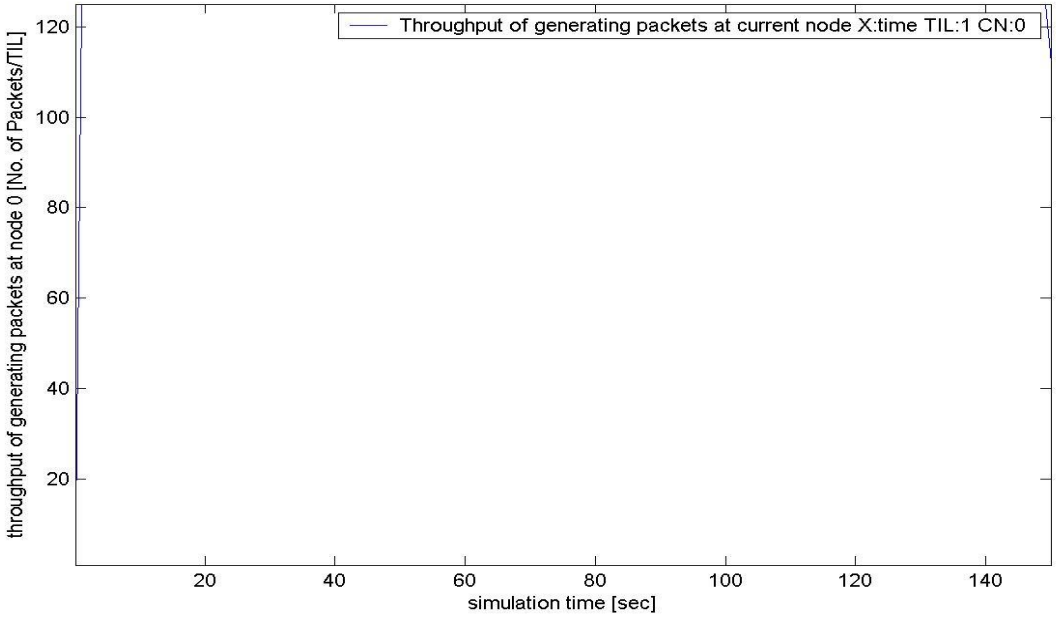

Fig. 4.19 Throughput of generating packets at node 0

Conclusion & Future Work

Conclusion

The GALS methodology removes the problem of arbitration delay. The global clocks and local clocks run at different time and the preventing the clock skew problem.

As the number of transistors is increasing, the NoC proves to be a handy method in overcoming the problems. But the communication clock and the computation clock should be kept separate, and for the same we use GALS methodology. This helps in removing the glitch that can be produced while making the SoC.

Clock of computation is related to two things:

- IP working
- Data Processing

Clock of communication is related to:

- Clock distribution within the task based clusters
- Clock distribution within the layer
- Clock distribution among the layers

Future Work

The future work will be to implement the NoC on a FPGA device. The GALS scheme will be used for the communication purposes.

References

[1] Xin Wang, “Designing Globally-Asynchronous Locally-Synchronous On-Chip Communication Networks”, Tampere 2008.

[2] Anoop Iyer, Diana Marculescu, “Power and Performance Evaluation of Globally Asynchronous Locally Synchronous Processors”, Pittsburgh.

[3] William James Dally, Principles and Practices of Interconnection Networks, San Francisco, Morgan Kaufmann, 2004.

[4] B. Nagaveni, G. Sai Thirumal, M. Rami Reddy, “Implementation of Network On-Chip Using GALS Scheme”, May 2013.

[5] Syed Suhaib, Deepak Mathaikutty, Sandeep Shukla, “Dataflow Architectures for GALS”, Blacksburg.

[6] G. Ascia, “Neighbours on path”, A new selection Strategy For On-chip Networks, in IEEE ESTIMedia, Hong Kong, 2006.